T-Coffee Documentation

Release Version_13.45.47.aba98c5

Cedric Notredame

Jun 23, 2021

Contents

Contents:

CHAPTER 1

T-Coffee Installation

Warning: This chapter has been extensively updated in 11/2016. Installation of T-Coffee versions lower than version 9.03 may now be deprecated. Contact us if you need to install an older version !

1.1 Installation

This chapter describes the installation procedures relevant for a standard use of T-Coffee on the most common operative systems: Unix/Linux and Mac OS. T-Coffee can only be installed on windows using virtual box simulating a Linux environement. We maintain two versions of T-Coffee:

1) the ** ‘stable version <http://tcoffee.org/Packages/tcoffee/Stable/> ‘** This is an extensively evaluated version, either linked to the latest paper or benefiting from an important fix- ** recommended **.

2) the ** ‘beta version <http://www.tcoffee.org/Packages/Beta/Latest/> ‘** is the latest release. It passes all the main tests and is usualy of higher quality that the stable. It is recommended to start installing this version if you experience any issue with the stable.

3) the ** ‘archives <http://www.tcoffee.org/Packages/Archives/> ‘** all previous releases of T-Coffee. If the one you need is missing please contact us.

T-Coffee is a complex package that interacts with many other third party software and/or servers (such as BLAST, see next section). All available versions on the server (starting from version 1.08) will install on your computer all the third party packages and setup the variables required for the different T-Coffee options to run correctly. Whenever the automated installation fails because of unforeseen system specificities, **don’t hesitate to contact us**. If the installation was successfull with the exception of some packages, users should install the third party package manually. This documentation gives some useful tips, but users are encouraged to send their feedbacks and share their experiences in order to improve it.

1.1.1 Unix/Linux Binaries

```
1. Download the installer package corresponding to your system from:
   http://tcoffee.org/Packages/Stable/Latest/

2. Grant execution permission to the downloaded file with the following command:
   ##: chmod +x T-COFFEE_installer_"version_x".bin

3. Launch the installation wizard with:
   ##: ./T-COFFEE_installer_"version_x".bin

4. Follow the wizard instructions and complete the installation

5. Open a new terminal session to be sure that your environment is updated

6. Type the following command to verify the installation was successful:
   $$: t_coffee -version
```

1.1.2 MacOS Binaries - Updated

```
##: Get the latest stable version from http://www.tcoffee.org/Packages/Stable/Latest/
##: Or  the latest Beta   Version from http://www.tcoffee.org/Packages/Beta/Latest/
##: download the *.tar.gz file
##: tar -xvf T-COFFEE_distribution_Version_XXXXX.tar.gz
##: cd T-COFFFEE_distribution_Version_XXXXX
##: ./install all
##: add the instructions given at the bottom of the install output to your .profile␣
↪or your .bashrc
```

1.1.3 Installation From Source/Binaries downloader (Mac OSX/Linux)

The following procedure shows howto install T-Coffee from the pre-packages source distribution. This procedure is recommanded because the version of T-Coffee available there is guarranteed to have passed a set of pre-specified tests. By contrast, the github version is untested, and therefore ongoing work... In most cases the binaries will be downloaded from a repository but you may also trigger local compilation.

```
1. Download the desired installer:
   http://www.tcoffee.org/Packages/Stable/Latest/T-COFFEE_distribution_Version_
↪<version>.tar.gz
   http://www.tcoffee.org/Packages/Beta/Latest/T-COFFEE_distribution_Version_<version>
↪.tar.gz
   http://www.tcoffee.org/Packages/Archives/T-COFFEE_distribution_Version_<version>.
↪tar.gz
   https://github.com/cbcrg/tcoffee

2a. untar the file
   ##: tar -xvf T-COFFEE_distribution.tar.gz
   ##: cd  T-COFFEE_distribution

2b. clone or unzip the github repo
   ##: cd t_coffee/src
```

(continues on next page)

(continued from previous page)

```
3a. Launch the installer
   ##:  ./install tcoffee


3b. If you want to use remote services (psiblast, expresso). You must enter a proxy␣
↪value and a valid e-mail. These values will be stored in <your home>/.t_coffee/.t_
↪coffee_env and can be edited at any time. The proxy is not compulsory. It is usualy␣
↪needed when working behind a firewall.
   ##: ./install tcoffee -email=<your email> -proxy=<your proxy>


4. as you can also install any of the T-Coffee mode you are interested in. Get the␣
↪list with:
   ##: ./install
   Then install one, for instance
   ##: ./install expresso
   Or install them all
   ##: ./install all


5. By default, the installer will not re-install any component already available on␣
↪your system. If you want to update you must specify
   ##: ./install <any component> -update
   OR
   ##: ./install <any component> -update -nobinaries
   For instance the following will cause T-Coffee to be recompiled on your system
   ##: ./install t_coffee -update -nobinaries

6. When you are done you will need to make this installation permanent by adding the␣
↪following lines *at the bottom* of your configuration file (typically .bashrc)
   ##: export PATH=<your home>/.t_coffee/bin/<macosx|linux>:$PATH
```

1.2 Template based modes: PSI/TM-Coffee and Expresso

The template modes are special modes of T-Coffee in whichj the tempates are associated with templates. The templates are easier to align the the sequences, thus resulting in more accurate alignments. The templates can be provided manually, or they can be fectched using BLAST. In order to do so, T-offee must be able to use BLAST. It can do so using a remote server like the EBI, or using your local installation.

BLAST is a program that searches databases for homologues of a query sequence. It works for protein and nucleic acid sequences alike. In theory BLAST is just a package like any but in practice things are a bit more complex. To run correctly, BLAST requires up-to-date databases (that can be fairly large, like nr or UniProt) and a powerful computer. Fortunately, an increasing number of institutes or companies are now providing BLAST clients that run over the net. It means that all you need is a small program that send your query to the big server and gets the results back. This prevents you from the hassle of installing and maintaining BLAST, but of course it is less private and you rely on the network and the current load of these busy servers.

Thanks to its interaction with BLAST, T-Coffee can gather more information and deliver alignments significantly more accurate than the default T-Coffee or any similar method. Let us go through the various modes available for T-Coffee...

1.2.1 Why do I need BLAST with T-Coffee?

The most accurate modes of T-Coffee scan the databases for templates that they use to align the sequences. Let's see how to get BLAST up and running, from the easy solution to tailored ones. There are currently two types of templates for proteins:

1) **structures**, that can be found by a BLASTP against the PDB database.

2) **profiles**, constructed using BLASTP or PSI-BLAST against nr or UniProt.

These templates are automatically built by T-Coffee when using one of the following modes:

```
To fetch and use structural templates:
##: t_coffee <yourseq> -mode expresso

To fetch and use profile templates:
##: t_coffee <your seq> -mode psicoffee

To fetch everything possible and get the best templates, structure or profile:
##: t_coffee <your seq> -mode accurate
```

Tip: BLAST based computation is a bit time intensive and results are cached to save time on re-computation (~/.t_coffee/cache/). These files are never erased so remember to empty the cache from time to time otherwise it's just getting bigger and bigger or use the option **-cache=no**

1.2.2 Using a BLAST local version on Unix

This is the most accurate way of using BLAST as it provides you with version control on both the program and the database. The downisde is that it requires installing BLAST and associated databases. If you simply want to make a quick try, you can use the remote cliant (see next section) If you have BLAST+ <https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download> installed, you can run it using the following command line:

:: ##: t_coffee <yourseq> -mode <expresso|piscoffee|tmcoffee> -blast_server=LOCAL -protein_db=<location of NR50 for psi/tm-coffee > -pdb_db=<location of PDB fasta database for expresso>

The default installation should be compliant with your system. At the time this section is being written (03/2020) Uniprot50 features about 36 million sequences and it takes about 3 minutes/sequences to build a profile on a mid-range workstation.

1.2.3 Using the EBI BLAST client

This is by far the easiest way and conveniently the default mode of T-Coffee. The PERL clients are already incorporated in T-Coffee and all you need are the proper PERL libraries. In principle, T-Coffee should have already installed these libraries during the standard installation, yet, this requires having root access. It really is worth the effort since the EBI is providing one of the best webservice available around and most notably, the only public PSI-BLAST via a webservice. Note that because PSI-BLAST is time consuming, T-Coffee stores the runs in its cache (**./tcoffee/cache**) so that it does not need to be rerun. It means that if you realign your sequences (or add a few extra sequences), things will be considerably faster.

Tip: The clients require the Perl module XML::Simple to be installed

Warning: Whenever you use a T-Coffee mode requiring BLAST access, it will ask you for an authentification e-mail. Do not provide fake e-mail, the EBI may suspend the service for all machines associated with your IP address (that could mean your entire lab, entire institute, even the entire country or, but I doubt it, the whole universe).

1.2.4 Using the NCBI BLAST client

The NCBI is the next best alternative however in my hands it was always a bit slower and, most of all, it does not incorporate PSI-BLAST as a webservice. A big miss! The NCBI web BLAST client is a small executable that you should install on your system. To do so, you just have to follow the instructions given on this link. Simply go for netbl, download the executable that corresponds to your architecture (Cygwin users should go for the win executable). Despite all the files that come along the executable blastcl3 is a stand alone executable that you can safely move to your $BIN. All you then need to do is to make sure that T-Coffee uses the right client; when you run T-Coffee, specify the client in the command line with the flag **-blast_server=NCBI**.

Tip: The clients require the Perl module XML::Simple to be installed

Attention: No need for any e-mail here, but you don't get PSI-BLAST. Whenever T-Coffee will need to use it, BLASTP will be used instead.

1.2.5 Using another client

You may have your own client (lucky you). If that is so, all you need is to make sure that this client is complient with the BLAST command line. If your client is named foo.pl, all you need to do is run T-Coffee command line with the flag **-blast_server=CLIENT_foo.pl**. Foo will be called as if it were BLASTPGP, and it is your responsability to make sure it can handle the following command line.

```
##: foo.pl -p <method> -d <db> -i <infile> -o <outfile> -m 7

"method"  : BLAST method for the search ("blastp" or "psiblast")
"db"      : database used for the search
"infile"  : input sequence(s) in FASTA format
"outfile" : name the output file
"-m 7"    : triggers the XML output (parses both the EBI & NCBI XML output)
```

Tip: If foo.pl behaves differently, the easiest way will probably be to write a wrapper around it so that wrapped_foo.pl behaves like BLASTPGP.

1.3 Troubleshooting

1.3.1 Third party packages

These procedures are not needed for default usage of T-Coffee. You will only need to install/configure these packages for specific purposes. T-Coffee is meant to interact with as many packages as possible, especially for aligning or using predictions. You will receive a list of supported packages that looks like the next table if you simply type **t_coffee**:

```
Command:
$$: t_coffee

Display the list of supported packages:

****** Pairwise Sequence Alignment Methods:
-------------------------------------------
fast_pair built_in
exon3_pair built_in
exon2_pair built_in
exon_pair built_in
slow_pair built_in
proba_pair built_in
lalign_id_pair built_in
seq_pair built_in
externprofile_pair built_in
hh_pair built_in
profile_pair built_in
cdna_fast_pair built_in
cdna_cfast_pair built_in
clustalw_pair ftp://www.ebi.ac.uk/pub/clustalw
mafft_pair http://www.biophys.kyoto-u.ac.jp/~katoh/programs/align/mafft/
mafftjtt_pair http://www.biophys.kyoto-u.ac.jp/~katoh/programs/align/mafft/
mafftgins_pair http://www.biophys.kyoto-u.ac.jp/~katoh/programs/align/mafft/
dialigntx_pair http://dialign-tx.gobics.de/
dialignt_pair http://dialign-t.gobics.de/
poa_pair http://www.bioinformatics.ucla.edu/poa/
probcons_pair http://probcons.stanford.edu/
muscle_pair http://www.drive5.com/muscle/
t_coffee_pair http://www.tcoffee.org
pcma_pair ftp://iole.swmed.edu/pub/PCMA/
kalign_pair http://msa.cgb.ki.se
amap_pair http://bio.math.berkeley.edu/amap/
proda_pair http://bio.math.berkeley.edu/proda/
prank_pair http://www.ebi.ac.uk/goldman-srv/prank/
consan_pair http://selab.janelia.org/software/consan/

****** Pairwise Structural Alignment Methods:
-------------------------------------------
align_pdbpair built_in
lalign_pdbpair built_in
extern_pdbpair built_in
thread_pair built_in
fugue_pair http://mizuguchilab.org/fugue/
pdb_pair built_in
sap_pair https://mathbio.crick.ac.uk/wiki/Software#SAP
mustang_pair http://lcb.infotech.monash.edu.au/mustang/
tmalign_pair https://zhanglab.ccmb.med.umich.edu/TM-align/

****** Multiple Sequence Alignment Methods:
-------------------------------------------
clustalw_msa ftp://www.ebi.ac.uk/pub/clustalw
mafft_msa http://www.biophys.kyoto-u.ac.jp/~katoh/programs/align/mafft/
mafftjtt_msa http://www.biophys.kyoto-u.ac.jp/~katoh/programs/align/mafft/
mafftgins_msa http://www.biophys.kyoto-u.ac.jp/~katoh/programs/align/mafft/
dialigntx_msa http://dialign-tx.gobics.de/
dialignt_msa http://dialign-t.gobics.de/
```

(continues on next page)

(continued from previous page)

```
poa_msa http://www.bioinformatics.ucla.edu/poa/
probcons_msa http://probcons.stanford.edu/
muscle_msa http://www.drive5.com/muscle/
t_coffee_msa http://www.tcoffee.org
pcma_msa ftp://iole.swmed.edu/pub/PCMA/
kalign_msa http://msa.cgb.ki.se
amap_msa http://bio.math.berkeley.edu/amap/
proda_msa http://bio.math.berkeley.edu/proda/
prank_msa http://www.ebi.ac.uk/goldman-srv/prank/

####### Prediction Methods available to generate Templates
------------------------------------------------------------
RNAplfold http://www.tbi.univie.ac.at/~ivo/RNA/
HMMtop http://www.enzim.hu/hmmtop/
GOR4 http://mig.jouy.inra.fr/logiciels/gorIV/
wublast_client http://www.ebi.ac.uk/Tools/webservices/services/wublast
blastpgp_client http://www.ebi.ac.uk/Tools/webservices/services/blastpgp
```

Tip: In our hands all these packages where very straightforward to compile and install on a standard Cygwin or Linux configuration. Just make sure you have gcc, the C compiler, properly installed. Once the package is compiled and ready to use, make sure that the executable is on your path, so that T-Coffee can find it automatically. Our favorite procedure is to create a bin directory in the home. If you do so, make sure this bin is in your path and fill it with all your executables (this is a standard Unix practice).

1.3.2 M-Coffee parameters

M-Coffee is a special mode of T-Coffee that makes it possible to combine the output of many Multiple Sequence Alignment packages. By default all the packages will be in the following folder **$HOME/.t_coffee/plugins/linux/**. If you want to have these packages in a different directory, you can either set the environment variable (option 1) or use the flag **-plugin** (to override every other setting). If for some reason, you do not want this directory to be on your path or you want to specify a precise directory containing the executables, you can use option 2. You can also set the following environment variables to the absolute path of the executable you want to use option 3: whenever they are set these variables will supersede any other declaration. This is a convenient way to experiment with multiple package versions. If you would rather have the mcoffee directory in some other location, set the MCOFFEE_4_TCOFFEE environement variable to the proper directory (option 4).

```
Option 1: set the environment variable
##: setenv PLUGINS_4_TCOFFEE=<plugins dir>

Option 2: specify the directory
##: export PLUGINS_4_TCOFFEE=<dir>

Option 3:
##: POA_4_TCOFFEE CLUSTALW_4_TCOFFEE TCOFFEE_4_TCOFFEE MAFFT_4_TCOFFEE \
MUSCLE_4_TCOFFEE DIALIGNT_4_TCOFFEE PRANK_4_TCOFFEE DIALIGNTX_4_TCOFFEE

Option 4:
##: setenv MCOFFEE_4_TCOFFEE <directory containing mcoffee files>
```

To be able to run M-Coffee, these following files are enough for a default usage:

```
BLOSUM.diag_prob_t10 BLOSUM75.scr blosum80_trunc.mat
dna_diag_prob_100_exp_330000 dna_diag_prob_200_exp_110000
BLOSUM.scr BLOSUM90.scr dna_diag_prob_100_exp_110000
dna_diag_prob_100_exp_550000 dna_diag_prob_250_exp_110000
BLOSUM75.diag_prob_t2 blosum80.mat dna_diag_prob_100_exp_220000
dna_diag_prob_150_exp_110000 dna_matrix.scr
```

1.3.3 Structural modes (using PDB)

Expresso/3D-Coffee are special modes of T-Coffee that allow to combine sequences and structures to reach more accurate alignments. T-Coffee proposes also other tools (iRMSD/APDB, T-RMSD, etc. . .) requiring access to structural information. You can do so either by having a database installed locally on your own system or by accessing the PDB through the web server. If you do not have PDB installed, don't worry, T-Coffee will go and fetch any structure it needs directly from the PDB repository, it will simply be a bit slower. If you prefer to have access to a local installation of the PDB in your file system, you have to indicate their location in your system using one of the following commands:

```
Using a local version of the PDB database:
##: setenv (or export) PDB_DIR <PATH>/data/structures/all/pdb/
##: setenv (or export) PDB_DIR <PATH>/structures/divided/pdb/
```

The T-RMSD tools comes along with T_Coffee package in order to build clustering based on structure. In addition to structural information it also requires the package Phylip, containing lots of phylogenetic tree reconstruction tools. If you need more information about the different Phylip tools, information can be obtained here.

1.3.4 R-Coffee associated packages

R-Coffee is a special mode able to align RNA sequences while taking into account their secondary structure. R-Coffee only requires the package Vienna to be installed, in order to compute Multiple Sequence Alignments. To make the best out of it, you should also have all the packages required by M-Coffee.

- Consan from Eddy/Riva laboratory.
- RNAplfold from the Vienna package.
- ProbConsRNA from Stanford university.

Tip: Regarding ProbConsRNA, make sure you rename the probcons executable into ProbConsRNA.

Tip: In order to insure a proper interface bewteen Consan and R-Coffee, make sure that the file mix80.mod is in the directory **~/.t_coffee/mcoffee** or in the mcoffee directory otherwise declared.

CHAPTER 2

Quick Start Regressive Algorithm

2.1 Introduction

This document introduces the regressive mode of T-Coffee (please cite the Nature Biotech original publication). It is meant to align very large datasets with a high accuracy. In order to ise it, you need a complete installation of T-Coffee with all supported third-party packages. These come by default in the installation package but you will have to install them manualy if you compile T-Coffee from its source code. See the installation section for more details.

Multiple sequence alignments (MSA) are usualy estimated by progressively aligning all the sequences, starting with the most similar. The regressive alignment is a new procedure that proceeds the other way around and starts by aligning the most distantly related sequences first. Like its progressive sibblin, the regressive algorithm starts with a guide tree. It uses this guide tree to extract the N most diverse sequences. In this first intermediate MSA, each sequence is either a leaf or the representative of a subtree. The algorithm is re-aplied recursively onto every representative sequence until all sequences have been incorporated in an internediate MSA of max size N. The final MSA is then obtained by merging all the intermediate MSAs into the final MSA. The merging is very efficient and does not require further alignment because the intermediate MSAs contain common sequences.

Aside from its improved accuracy, the main benefit of this protocol is to allow any third party multiple sequence aligner to be applied on the sub-groups (ClustalO, MAfft, etc) and any third party guide-tree protocol. The default T-Coffee supports a large number of internal and external package but if the one you need is not supported, note that it is relatively simple to create a configuration file allowing T-Coffee to interact with the package you need as docimented here.

When deployed serialy, the regressive algorithm scales linearly with the number of sequences both in time and memory. The amount of memory it needs is roughly the amount of memory required to align N sequences with the selected alignment method. Given TOT sequences, the amount of time is roughly TOT/N*(amount of time required to align N Seq with the selected method). The algorithm implementation is multi-threaded and owing to its embarrasingly parralel nature, it benefits from a very efficient speedup when deployed this way.

In the following document, we list its main options and how these can be deployed from the T-Coffee algorithm.

Note: REG does not support ALL the T-Coffee options. It is especially limited with respect to input and output and will only allow FASTA.

```
##: Get the latest stable version from http://www.tcoffee.org/Packages/Stable/Latest/
##: Or  the latest Beta   Version from http://www.tcoffee.org/Packages/Beta/Latest/
##: download the *.tar.gz file
##: tar -xvf T-COFFEE_distribution_Version_XXXXX.tar.gz
##: cd T-COFFFEE_distribution_Version_XXXXX
##: ./install all
##: add the instructions given at the bottom of the install output to your .profile␣
↪or your .bashrc
```

2.2 Installation from source

```
##: get the latest version from https://github.com/cbcrg/tcoffee
##: cd tcoffee-master/t_coffee/src
##: make t_coffee
##: mv t_coffee <Binary Directory>
```

In order to use T-Coffee you must also have the following pacaked installed

```
## Mafft:              https://mafft.cbrc.jp/alignment/software/
## ClustalOmega:       http://www.clustal.org/omega/#Download
```

More sophisticated modes of the regressive algorithm that use either profile extension (psicoffee) or structural templates (3dcoffee or Expresso)require a local installation of Blast+ and associated databases

```
## Blast+:             https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/
## Uniprot50:          https://www.uniprot.org/downloads
## PDB Fasta:          ftp://ftp.wwpdb.org/pub/pdb/derived_data/pdb_seqres.txt.gz
```

Whenever structural modes are used, T-Coffee is able to download directly the PDBs from rscb. It is therefore not needed to install PDB locally, but the program can also use a local version of PDB. T-Coffee is, however, able to use a local version installed in a standard way (cf the RSCB website: https://www.rcsb.org/)

2.3 Examples

2.3.1 Fast and accurate

In the following example, the regressive alignment is used to align the sequences in FASTA format. The tree is estimated using the mbed method of Clustal Omega (-reg_tree=mbed), the size of the groups is 100 (-reg_nseq=100) and the method used to align the groups is Clustal Omega:

```
$$: t_coffee -reg -seq proteases_large.fasta -nseq 100 -tree mbed -method clustalo_
↪msa -outfile proteases_large.aln -outtree proteases_large.mbed
```

This mode is the default mode and is expected to deliver fast and reasonnably accurate alignments

2.3.2 Slower and more accurate

This mode is expected to be the most accurate and combines the ginsi mode of MAFFT which is very accurate with the mbed trees of Clustal Omega. Note that it is the regressive deployment that allows ginsi to be used on datasets of unlimited size.

```
$$: t_coffee -reg -seq proteases_large.fasta -nseq 100 -tree mbed -method mafftginsi_
↪msa -outfile proteases_large.aln -outtree proteases_large.mbed
```

2.3.3 Very Fast

This mode is expected to be the fastest currently available. Its accuracy is comparable to that of MAFFT-fftnsi running on its own

```
$$: t_coffee -reg -seq proteases_large.fasta -nseq 100 -tree parttree -reg_method␣
↪mafftfftnsi_msa -outfile proteases_large.aln -outtree proteases_large.parttree
```

2.4 Regressive flags

- **-seq** (usage:**-seq=<FASTA sequence file>**)

This flag is mandatory to provide the sequences. The sequences must be in FASTA and must not contain any sequences. Because these sequences are meant to be aligned with third party aligners that may not support the IUPAC extended alphabet, it is advisable to avoid non standard amino-acids symbols. It is also not advisable to provide gapped sequences.

- **-tree** (usage:**-tree=<method or Newick File>**, default: **mbed**)

This flag defines which method will be used to estimate the tree. The following methods are available

```
Tree Computation Method:
- mbed        : use mBed mode of ClustalO - Default
- cwdnd       : use the quicktree mode of ClustalW
- parttree    : parttree method of MAFFT - fastest option. Does not support sequences␣
↪less than 6 AA long
- dpparttree  : MAFFT fast clustering method
- fastparttree: MAFFT fast clustering method
- mafftdnd    : default MAFFT NJ tree - slower than the parttree modes
- fftns1dnd   : Tree produced after the first iteration MAFFT fftns mode
- fftns2dnd   : Tree produced after the second iteration MAFFT fftns mode
- upgma       : upgma tree - warning cubic time computation
- nj          : Neighbour Joinning tree
- famsadnd    : FAMSA single linkage tree
- #<command>  : Runs comamnd <seq> > <tree>.
- filename    : Any file in newick format. The seq file and the tree file must match
```

- **-newtree** (usage:**-newtree=<filename>** , default: <infile>.<reg_tree>)

This flag defines the name of the newly computed ouput tree. Deafult will be filename.reg_tree

- **-child_tree** (usage:**-chile_tree=<tree_method>** , default:None)

This flag defines which method will be used to estimate the tree. Note that by default, each <method> will use its own default tree mode.

- **-outfile** (usage:**-outfile=<filename>** , default: <infile>.aln)

This flag defines the name of the output file containing the multiple sequence alignment

- **-nseq** (usage:**-nseq=N** , default: 1000 for datasets larger than 10,000 and Nseq/10 for smaller datasets)

Sets the maximum size of the subsequence alignments. The recommanded value is 1000. With slow/accurate aligners that do not scale in a linear way, this parameter can have an importnat impact on CPU requirement with small values resulting in faster computation.

- **-method** (usage:**-method=<method or configuration file>** , default: clustalo_msa)

This flag defines which method will be used to estimate the child MSA. In order to know which methods are available.

```
$$: t_coffee
```

All methods the multiple sequence alignment methods xxx_msa are supported.

- **-thread** (usage:**-thread=<N> , default: 1, 0 causes all available procs to be used)

This flag defines how many procs will be used. Children MSAs are computed in parrallel

- **-dynamic** (usage:**-dynamic=<Integer>** , default: 1)

This flag defines the factor by which every Child reg_n is increased while going from root to leaf. By default all children contain the same maximum number of sequences, if dynamic>1, this number is set to reg_nseq*(dynamic^NGeneration from root). If dynamic <1, this number is set to: reg_nseq/(-dynamic^NGeneration from root). *Note that this mode has not bee validated*

- **-dynamic_config** (usage:**-dynamic_config=<file>** , default: none)

This flag provides a file specifying which method will be used depending on Nseq in slaves when using dynamic_msa. The format is

```
<method>       <maxnseq>
```

The default is

```
psicoffee_msa 20
famsa_msa     100000000
```

All methods (*_msa) available in t_coffee are supported.

- **-thread** (usage:**-nseq=N** , default: 1, 0 to use all available threads)

Sets the maximum number of threads to be used by one instance. Note that if you have scrited your own aligner using a configuration file, you should make sure it does not run in a multi-threaded version as well.

- **-child_thread** (usage:**-child_thread=<N> , default: 1, 0 causes all available procs to be used)

This flag defines how many procs will be allocated to the computation of *each* child MSA. This makes sense when the children MSAs are very large

If you want to use an non-supported method, follow these guidelines.

CHAPTER 3

Quick Start Unistrap

3.1 Introduction

This document introduces the unsitsrap strategy that incorpates MSA instability into the estimation of the Felsentein Bootstrap replicates (please cite the Systematic Biology original publication).

The principke is very straightorward. It involves shuffling the input order of the sequences so as to generate MSA shuffle replicates, from which columns are then drawn with replacement so as to generate the Bootsrap replicates. As established in the paper, it is also possible to shuffle left and right chidren in the guide tree of any progressive methods. This procedure recapitulates two third of the instability measured when shuffling sequences. Whenever using a progressive algorithm, the tree shuffling procedure should be applied in order to by-pass any arbitrary stabilisation procedures (internal sorting for instance).

The procedure used for the original validation is available on Github). The content of this repository is not especially user friendly and its sole purpose is to allow reproducing the analysis provided in the original paper. For practical usage, we recommand using the re-implementation provided below that can be accessed using the T-Coffee package. This re-implementation supports all the T-Coffee supported aligners when shuffling input sequences (**t_coffee ** to get a list) and a subset of progressive aligners when using the tree shuffle strategy.

Aside from its improved accuracy, the main benefit of this protocol is to allow any third party multiple sequence aligner to be applied on the sub-groups (ClustalO, MAfft, etc) and any third party guide-tree protocol. The default T-Coffee supports a large number of internal and external package but if the one you need is not supported, note that it is relatively simple to create a configuration file allowing T-Coffee to interact with the package you need as docimented here.

When deployed serialy, the regressive algorithm scales linearly with the number of sequences both in time and memory. The amount of memory it needs is roughly the amount of memory required to align N sequences with the selected alignment method. Given TOT sequences, the amount of time is roughly TOT/N*(amount of time required to align N Seq with the selected method). The algorithm implementation is multi-threaded and owing to its embarrasingly parralel nature, it benefits from a very efficient speedup when deployed this way.

In the following document, we list its main options and how these can be deployed from the T-Coffee algorithm.

Note: REG does not support ALL the T-Coffee options. It is especially limited with respect to input and output and

will only allow FASTA.

```
##: Get the latest stable version from http://www.tcoffee.org/Packages/Stable/Latest/
##: Or  the latest Beta   Version from http://www.tcoffee.org/Packages/Beta/Latest/
##: download the *.tar.gz file
##: tar -xvf T-COFFEE_distribution_Version_XXXXX.tar.gz
##: cd T-COFFFEE_distribution_Version_XXXXX
##: ./install all
##: add the instructions given at the bottom of the install output to your .profile␣
↪or your .bashrc
```

3.2 Examples

3.2.1 Produce bootsrap replicates by shuffling sequences and guide tree sister nodes

In the following example, the input order of the sequences will be shuffled 10 times so as to generate as many MSA replicates (-msa). Each time a guide tree will be generated using mbed (-tree), and the sister nodes of this tree will be randomly shuffled (i.e. left and right will be randoml?y reassigned to left or right) - note that this shuffling does not change the topology. For each shuffled replicated, an MSA willl be estimated with clustalo (-method) and each MSA this produced will be used to produce 10 replicates.

Note: Note: This mode will re-compute a guide tree for every sequence shuffle. The tree shuffle is not needed but it will increase the probability of each iteration delivering a different MSA. It is much more computationaly efficient to replace the sequence order shuffle with a tree shuffle thgat will generate a comparable amount of MSA diversity at a significantly lower computational cost.

```
$$: t_coffee -other_pg unistrap -in sample_seq1.fasta -tree mbed -method clustalo -
↪shuffle_tree -shuffle_seq -msa 10 -bootstrap 10 -outfile replicates.phylip
```

3.2.2 Produce bootsrap replicates by shuffling the guide tree sister nodes

In the following example, a guide tree will be generated once using mbed (-tree), and the sister nodes of this tree will be randomly shuffled (i.e. left and right will be randoml?y reassigned to left or right) 10 times. For each shuffled replicated, an MSA willl be estimated with clustalo (-method) and each MSA this produced will be used to produce 10 replicates.

```
$$: t_coffee -other_pg unistrap -in sample_seq1.fasta -tree mbed -method clustalo -
↪shuffle_tree -msa 10 -bootstrap 10 -outfile replicates.phylip
```

3.2.3 Get list of supported methods

The following command will list all the guide tree methods and multiple aligners supported by unistrap

```
$$: t_coffee -other_pg unistrap  -tree list -method list
```

3.3 unistrap flags

- **-in** (usage:**-seq=<FASTA sequence file>**)

This flag is mandatory to provide the sequences. The sequences must be in FASTA and must not contain any sequences. Because these sequences are meant to be aligned with third party aligners that may not support the IUPAC extended alphabet, it is advisable to avoid non standard amino-acids symbols. It is also not advisable to provide gapped sequences.

- **-shuffle_tree** (usage:**-shuffle_tree**, default: **on**)

This flag defines the shuffling of sister nodes for each MSA replicate. The guide tree does not need to be re-estimated

- **-shuffle_seq** (usage:**-shuffle_seq**, default: **off**)

This flag defines the shuffling of input sequences for each MSA replicate. The guide tree *must be re-estimated*.

- **-tree** (usage:**-tree=<method,filename,list>** , default: mbed)

This flag defines the method used to estimate the guide tree, it can be a method or an existing tree

- **-method** (usage:**-tree=<method,list>** , default: clustalo)

This flag defines the method used to estimate the MSA.

- **-msa** (usage:**-msa=<integer>** , default: 10)

This flag defines the number of MSA replicates

- **-bootstrap** (usage:**-bootstrap=<integer>** , default: 10)

This flag defines the number of Bootsrap replicates. Note that the final number of replicates will be msa* x bootstrap

- **-outfile** (usage:**-outfile=<tree_method>** , default:stdout)

This flag defines the outfile name

CHAPTER 4

Quick Start T-Coffee

Warning: This chapter has been extensively updated in 11/2016. All T-Coffee modes/tools/commands should be valid for version 9.03 and above, but it is not guaranteed for lower versions of T-Coffee.

4.1 Basic Command Lines (or modes)

This chapter is a quick overview on how to run T-Coffee alignment using predefined procedures we call "modes". All the files mentioned can be found here. You can also use examples associated with their corresponding command lines from the subsection **Tutorial (practical examples)** published in Nature Protocols (2011). Refer to the section **T-Coffee Manual** for more technical details about T-Coffee usage and tools. Please, use the corresponding citation when using a specific mode or tools of T-Coffee (see **References and Citations**) otherwise cite T-Coffee.

4.1.1 Protein sequences

```
------------------------------------------------------------------------------
Default
------------------------------------------------------------------------------

$$: t_coffee sample_seq1.fasta

Citation: Notredame et al., JMB (2000)                      PMID:10964570
------------------------------------------------------------------------------
Fast
------------------------------------------------------------------------------

$$: t_coffee sample_seq1.fasta -mode quickaln

Citation: Notredame et al., JMB (2000)                      PMID:10964570
------------------------------------------------------------------------------
Consistent (M-Coffee combines the most common MSA packages)
```

(continues on next page)

(continued from previous page)

```
------------------------------------------------------------------------------

$$: t_coffee sample_seq1.fasta -mode mcoffee

Citation: Wallace et al., Nucleic Acids Res. (2006)         PMID:16556910
------------------------------------------------------------------------------
Structure (Expresso finds structures homologous to your sequences)
------------------------------------------------------------------------------

$$: t_coffee sample_seq1.fasta -mode expresso

Citation: Armougom et al. Nucleic Acids Res. (2006)         PMID:16845081
------------------------------------------------------------------------------
Homology (PSI-Coffee enriches your dataset with homologous sequences)
------------------------------------------------------------------------------

$$: t_coffee sample_seq1.fasta -mode psicoffee

Citation: Chang et al., BMC Bioinformatics (2012)           PMID:22536955
------------------------------------------------------------------------------
Accurate (combines Structures and Homology)
------------------------------------------------------------------------------

$$: t_coffee sample_seq1.fasta -mode accurate

Citation: Notredame et al., JMB (2000)                      PMID:10964570
```

4.1.2 DNA sequences

```
------------------------------------------------------------------------------
Default
------------------------------------------------------------------------------

$$: t_coffee sample_dnaseq1.fasta

Citation: Notredame et al., JMB (2000)                      PMID:10964570
------------------------------------------------------------------------------
Functional (Pro-Coffee increases accuracy of functional DNA regions )
------------------------------------------------------------------------------

$$: t_coffee sample_dnaseq1.fasta -mode procoffee

Citation: Erb et al., Nucleic Acids Res. (2012)             PMID:22230796
```

4.1.3 RNA sequences

```
------------------------------------------------------------------------------
Default
------------------------------------------------------------------------------

$$: t_coffee sample_rnaseq1.fasta
```

(continues on next page)

(continued from previous page)

```
Citation: Notredame et al., JMB (2000)                    PMID:10964570
-------------------------------------------------------------------------
Structure 2D (R-Coffee uses predicted secondary structures)
-------------------------------------------------------------------------

$$: t_coffee sample_rnaseq1.fasta -mode rcoffee

Citation: Wilm et al., Nucleic Acids Res. (2008)          PMID:18420654
-------------------------------------------------------------------------
Structure 3D (R-Coffee combined with Consan structural alignments)
-------------------------------------------------------------------------

$#: t_coffee sample_rnaseq1.fasta -mode rcoffee_consan

Citation: Wilm et al., Nucleic Acids Res. (2008)          PMID:18420654
-------------------------------------------------------------------------
Accurate (RM-Coffee use M-Coffee and secondary structure predictions)
-------------------------------------------------------------------------

$$: t_coffee sample_rnaseq1.fasta -mode rmcoffee

Citation: Wilm et al., Nucleic Acids Res. (2008)          PMID:18420654
```

4.2 Brief Overview of T-Coffee Tools

We only give you the very basics here, please go to the **T-Coffee Main Documentation** for a more detailed description. You can also try the **T-Coffee tutorial** for a practical training on T-Coffee alignment and other tools using applied examples on published research data.

4.2.1 Alignment methods

4.2.1.1 T-Coffee

Write or copy all your sequences (protein, DNA or RNA) in a given text file using one of the following format: Swiss-Prot, FASTA or PIR. Run T-Coffee with the following command line:

```
$$: t_coffee sample_seq1.fasta
```

When aligning, T-Coffee will always at least generate three files:

- `sample_seq1.aln` : Multiple Sequence Alignment (ClustalW format by default)
- `sample_seq1.dnd` : guide tree (Newick format)
- `sample_seq1.html` : colored MSA according to consistency (html format)

In principle, the type of the sequences is automatically detected and the default methods adapted accordingly. Sometimes, however, this may fail either because the sequences are too short or contain too many ambiguity codes. When this happens, you are advised to explicitly set the type of your sequences using the flag **-type**.

```
$$: t_coffee sample_dnaseq1.fasta -type=dna
```

Note: Please cite: Notredame, C., Higgins, D.G., Heringa, J. T-Coffee: a novel method for fast and accurate multiple sequence alignment. J. Mol. Biol., 302(1):205-217 (2000), PMID:10964570 and/or Magis, C., Taly, J.-F., Bussotti, G., Chang, J.M., Di Tommaso, P., Erb, I., Espinosa-Carrasco, J., Notredame, C. **T-Coffee: tree-based consistency objective function for alignment evaluation**. Methods Mol. Biol., 1079:117-129 (2014), PMID:24170398

4.2.1.2 M-Coffee

M-Coffee is a meta version of T-Coffee that combines the output of eight aligners (MUSCLE, ProbCons, POA, DIALIGN-T, MAFFT, ClustalW, PCMA and T-Coffee); when installing T-Coffee, all required packages are automatically installed on your computer. To use M-Coffee, write your sequences in a file (format: Swiss-Prot, FASTA or PIR) and run the following command 1. M-Coffee is a predefined combination of different types of aligners; there is a faster version called fm-Coffee (command 2) which combines the fastest aligners (Kalign, MUSCLE and MAFFT). Finally, the user can make its own combination of aligners included in T-Coffee by specifying the list of packages to be combined; here is an example of T-Coffee combining ClustalW, Kalign and ProbCons (command 3).

```
Command 1: running M-Coffee
$$: t_coffee sample_seq1.fasta -mode mcoffee

Command 2: running fm-Coffee
$$: t_coffee sample_seq1.fasta -mode fmcoffee

Command 3: user defined multiple methodes
$$: t_coffee sample_seq1.fasta -method clustalw_pair, kalign_pair, probcons_pair
```

Warning: If the program starts complaining one package or the other is missing, this means you will have to go the hard way and install all these packages yourself...

Note: Please cite: Wallace, I.M., O'Sullivan, O., Higgins, D.G., Notredame, C. **M-Coffee: combining multiple sequence alignment methods with T-Coffee**. Nucleic Acids Res., 34(6):1692-1699 (2006), PMID:16556910

4.2.1.3 Expresso

The default installation of T-Coffee provides you with the EBI wublast.pl client required to run Expresso) command 1). Using this, Expresso will BLAST your sequences against the PDB database, identify the best targets (by default X-RAY structures, minimum 35% identical to your sequences) and use them to align your proteins using a structural aligner. If all the required structural packages for Expresso are not installed or if you want to select another structural aligner, you can select the structural package you want to use, for instance, if can use TM-align rather than SAP (command 2).

```
Command 1:
$$: t_coffee sample_seq1.fasta -mode expresso

Command 2:
$$: t_coffee sample_seq1.fasta -template_file PDB -method TMalign_pair
```

This correspondence between sequences and structures (templates) is declared in a FASTA-like file we call template file. Expresso automatically generates the template file (`<your file name>_pdb1.template_list`) that can

be reused for applications, but you can also provide your own with the following format. This template file should have the following format:

```
> <seq_name> _P_ <PDB structure file or name>

******* sample_3Dseq1.template *******
>TNFR10-2  _P_ 1D4V2.pdb
>TNFR10-3  _P_ 1D4V3.pdb
...
**************************************
```

Note: Please cite: Armougom, F., Moretti, S., Poirot, O., Audic, S., Dumas, P., Schaeli, B., Keduas, V., Notredame. C. **Expresso: automatic incorporation of structural information in multiple sequence alignments using 3D-Coffee**. Nucleic Acids Res., 34:W604-W608 (2006), PMID:16845081

4.2.1.4 R-Coffee

R-Coffee can be used to align RNA sequences, using their RNApfold predicted secondary structures (command 1). The best results are obtained by using the Consan pairwise method. If you have Consan installed (under maintenance...), you get access to one of the most accurate mode of R-Coffee (command 2). This will only work if your sequences are short enough (less than 200 nucleotides). A good alternative is the rmcoffee mode (command 3) that will run MUSCLE, ProbCons4RNA and MAFFT and then use the secondary structures predicted by RNApfold. Finally, you can also select yourself which methods should be combined by R-Coffee (command 4).

```
Command 1: R-Coffee
$$: t_coffee sample_rnaseq1.fasta -mode rcoffee

Command 2: R-Coffee + Consan
$#: t_coffee sample_rnaseq1.fasta -mode rcoffee_consan

Command 3: RM-Coffee
$$: t_coffee sample_rnaseq1.fasta -mode rmcoffee

Command 4: user defined R-Coffee
$$: t_coffee sample_rnaseq1.fasta -mode rcoffee -method lalign_id_pair,slow_pair
```

Note: Please cite: Wilm, A., Higgins, D.G., Notredame, C. **R-Coffee: a method for multiple alignment of non-coding RNA**. Nucleic Acids Res., 36(9):e52 (2008), PMID:18420654

4.2.1.5 Pro-Coffee

Pro-Coffee is a particular mode of T-Coffee designed to align specific functional DNA sequences, in particular regulatory regions. To run Pro-Coffee by default, just use command 1. In order to adjust the quality of the alignment, Pro-Coffee allows you to modify gap penalties (gap-opening and/or gap-extension) with specific flags (command 2).

```
Command 1: Pro-Coffee default
$$: t_coffee sample_dnaseq1.fasta -mode procoffee

Command 2: Pro-Coffee with modified parameters
$$: t_coffee sample_dnaseq1.fasta -method promo_pair@EP@GOP@-60@GEP@-1
```

Note: Please cite: Erb, I., González-Vallinas, J.R., Bussotti, G., Blanco, E., Eyras, E., Notredame, C. **Use of ChIP-Seq data for the design of a multiple promoter-alignment method**. Nucleic Acids Res., 40(7):e52 (2012), PMID:22230796.

4.2.2 Evaluation tools

4.2.2.1 TCS (MSA evaluation based on consistency)

Transitive Consistency Score (TCS) is an alignment evaluation score that makes it possible to identify the most correct positions in an MSA. It has been shown that these positions are the most likely to be structuraly correct and also the most informative when estimating phylogenetic trees. The TCS evaluation and filtering procedure is implemented in the T-Coffee package and can be used to evaluate and filter any third party MSA (including T-Coffee MSA of course!).

It's usage is a bit tricky as it comes with a lot of different options, go to the **T-Coffee Main Documentation**, section **Evaluating Your Alignment** to have all the details about TCS.

Note: Please cite: Chang, J.-M., Di Tommaso, P., Notredame, C. **TCS: A new multiple sequence alignment reliability measure to estimate alignment accuracy and improve phylogenetic tree reconstruction**. Mol. Biol. Evol., 31(6), 1625–1637 (2014), PMID:24694831 and/or Chang, J.-M., Di Tommaso, P., Lefort, V., Gascuel, O., Notredame, C. **TCS: a web server for multiple sequence alignment evaluation and phylogenetic reconstruction**. Nucleic Acids Res., 43(W1):W3-6 (2015), PMID:25855806

4.2.2.2 iRMSD/APDB (MSA structural evaluation)

iRMSD/APDB is not an alignment tool, it is an evalution tool of a given alignment using structural information. All you need is a file containing the alignment of sequences with a known structure ("template file"; see Expresso). If you don't provide a template file, these sequences must be named according to their PDB ID, followed by the chain index (1aabA for example). In the first example (command 1) names are different therefore it won't deliver any result. In that case, you should declare the correspondence between sequences and structures using your own template file (command 2). All the sequences do not need to have a known structure, but at least two is required otherwise it won't deliver any result.

```
Command 1:
$$: t_coffee -other_pg irmsd sample_3Dseq1.aln

Command 2:
$$: t_coffee -other_pg irmsd sample_3Dseq1.aln -template_file sample_3Dseq1.template
```

Note: Please cite: Armougom, F., Moretti, S., Keduas, V., Notredame, C. **The iRMSD: a local measure of sequence alignment accuracy using structural information**. Bioinformatics, 22(14):e35-e39 (2006), PMID:16873492

4.2.2.3 STRIKE (single structure MSA evaluation)

Under maintenance on the webserver or the T-Coffee package...

4.2.2.4 T-RMSD (structural clustering)

T-RMSD is a structure based clustering method using the iRMSD to drive the structural clustering of your aligned sequences with an available structure. The T-RMSD supports all the parameters supported by iRMSD or APDB. To run T-RMSD, type:

```
$$: t_coffee -other_pg trmsd sample_3Dseq1.aln -template_file sample_3Dseq1.template
```

The program then outputs a series of files:

- `sample_3Dseq1.struc_tree.list` : list of the trees associated with every position.
- `sample_3Dseq1.struc_tree.html` : colored columns supporting the tree.
- `sample_3Dseq1.struc_tree.consensus_output` : schematic display of the results.
- `sample_3Dseq1.struc_tree.consensus` : final consensus structural tree.

Note: Please cite: Magis, C., Stricher, F., van der Sloot, A.M., Serrano, L., Notredame, C. **T-RMSD: a fine-grained, structure based classification method and its application to the functional characterization of TNF receptors**. J. Mol. Biol., 400(3):605-617 (2010), PMID:20471393 and/or Magis, C., van der Sloot, A.M., Serrano, L., Notredame, C. **An improved understanding of TNFL/TNFR interactions using structure-based classifications**. Trends Biochem. Sci., 37(9):353-363 (2012), PMID:22789664

4.3 Tutorial (Practical Examples)

Note: This documentation is merely a cheat-sheet that recapitulates the material and the command lines associated with the manual. This tutorial itself is adpated from the T-Coffee Nature Protocols Article that can be followed step by step on the following website

---

4.3.1 Introduction

T-Coffee is a versatile Multiple Sequence Alignment method suitable for aligning most types of biological sequences. The series of protocols presented here show how the package can be used to multiply align proteins, DNA and RNA sequences. The package is an open source freeware available from our website.

There are several parts: 1) the protein section presents controlled cases for PSI-Coffee the homology extended mode suitable for remote homologues, Expresso the structure based multiple aligner and M-Coffee, a meta version able to combine several third party aligners into one, 2) we then show how the T-RMSD option can be used to produce a functionally informative structure based clustering, 3) RNA alignment procedures are shown for R-Coffee a mode that produces secondary structure based MSAs, 4) DNA alignments are illustrated with Pro-Coffee, a multiple aligner specific of promoter regions, 5) finally, the last section presents some of the many reformatting utilities bundled with T-Coffee.

4.3.2 Materials

The list of files (input and output) required by this protocol is available from here. They can be automatically retrieved using the following command:

```
$$: t_coffee -other_pg nature_protocol.pl
```

This will create 4 repertories containing the input sequences necessary for the protocols we report in this section. For each part, all command lines have been collected into the file README.sh.

4.3.3 Procedures

- Full Tutorial
- Installation
- Protein Multiple Sequence Alignments
- RNA Multiple Sequence Alignments
- Promoter alignments
- Reformat alignments

CHAPTER 5

T-Coffee Main Documentation

Warning: This chapter has been extensively updated in 11/2016. We improved its readibility and incorporate all the latest development of the past years (PSI/TM-Coffee for homology extension based MSAs, Pro-Coffee for functional DNA, R-Coffee/SARA-Coffee for RNA, TCS and STRIKE for evaluating MSAs, T-RMSD for structural clustering). If you have any suggestion or correction, don't hesitate to contact us.

5.1 Before You Start...

5.1.1 Foreword

Most of the work presented here emanates from two summer schools that were tentatively called the 'Prosite Workshops' and were held in Marseille, in 2001 and 2002. These workshops were mostly an excuse to go rambling and swimming in the creeks of Cassis (Calanques). Yet, when we got tired of lazing in the sun, we eventually did a bit of work to chill out. Most of our experiments were revolving around the development of sequence analysis tools. Many of the most advanced ideas in T-Coffee were launched during these fruitful sessions. Participants included Phillip Bucher, Laurent Falquet, Marco Pagni, Alexandre Gattiker, Nicolas Hulo, Christian Siegfried, Anne-Lise Veuthey, Virginie Leseau, Lorenzo Ceruti and Cedric Notredame.

5.1.2 Prerequisite for using T-Coffee

This documentation relies on the assumption that you have installed T-Coffee, version 9.03 or higher (preferably the latest stable version). T-Coffee is a freeware open source running on all Unix-like platforms including Mac OS X, woever, it cannot run on Windows except by using Cygwin, a freeware open source allowing to run a Unix-like command line on Windows (download). A better option for Windows users would be to install a Unix-like virtualbox on your computer. All the relevant information for installing T-Coffee is contained in the previous chapter **T-Coffee Installation**.

5.1.3 Let's have a try...

This manual is made to help you discover (nearly) all subtleties of T-Coffee, from default applications to sophisticated ones. All along this documentation, we expect you to use Unix-like shell commands running on prepared example files we have created. For now you can find all these files on our github repository for now: example files; however in the future they will be included in the T-Coffee distribution (~/tcoffee/Version.../examples/). All T-Coffee commands indicated with a tag "$$:" are currently up and running; just copy/paste these commands from the documentation (without the tag "$$:") to your terminal provided you previously downloaded the example files. We encourage you to try all these examples which are controlled cases, and also to try these commands with your own data files. All T-Coffee commands indicated with a tag "$#:" are currently under maintenance. Also, be aware that all the values/results given in this manual were obtained with different versions of T-Coffee; depending on the version of T-Coffee you are using, these results may slightly differ.

Important: Using T-Coffee package via "command lines" is the best/only way **if you need to do something sophisticated and/or highly reproducible**, but if you don't want to bother with command line you can use our web server or different links on the Cedric Notredame's lab homepage. We have tried to put as many of T-Coffee functionalities on the webserver and we plan to incorporate even more in the future, but for now some options are not available or limited.

5.2 What is T-Coffee ?

5.2.1 What is T-Coffee?

T-Coffee stands for **"Tree based Consistency Objective Function For alignment Evaluation"**. It is mainly/primarily a Multiple Sequence Alignment (MSA) method but it also provides a collection of useful/powerful tools presented in this manual. Before going deep into the core of the matter, here are a few words to briefly explain the things T-Coffee can do !!!

5.2.1.1 What does it do?

T-Coffee has two components allowing you to perform different tasks:

- mainly it is a Multiple Sequence Alignment method. Given a dataset of sequences previously gathered using database search programs (like BLAST, Ensembl, etc...), T-Coffee will produce a Multiple Sequence Alignment (MSA) (refer to the section **Building Your Multiple Sequence Alignments** for more details).
- also it is a collection of tools (usually called with the flag **-other_pg "tools"**) and third party software (cf. section **Integrating External Methods in T-Coffee**) able to perform a wide range of operations such as aligning accurately, reformatting data, evaluating results, comparing methods and many more...

5.2.1.2 What can it align?

T-Coffee will align nucleic acid (DNA and RNA) and protein sequences alike. T-Coffee is also able to use other type of information such as secondary/tertiary structure information (for protein or RNA sequences with a known/predicted structure), sequence profiles, trees...

Hint: To have a rough idea, on an average computer, T-Coffee can easily align up to a 200 sequences, about 1000 amino acid long in about ~20min. It is better to avoid aligning more than 1000 sequences without using a cluster or T-Coffee fast modes !!!

5.2.1.3 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 and syntax. If you do not like typing commands, you can still use the latest T-Coffee webserver here.

Tip: 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.

5.2.1.4 Is there an online webserver?

Yes, at Cedric Notredame's lab homepage which contains all the necessary links to our latest webserver !

5.2.1.5 Is T-Coffee different from ClustalW?

According to several benchmarks, T-Coffee is on overall much more accurate than ClustalW, but this increase in accuracy comes at a price: **T-Coffee (default mode) is slower than ClustalW** (about N times for N Sequences). Still, if you want to align closely related sequences, **T-Coffee can be used in a fast mode, much faster and about as accurate than ClustalW** (cf. section **Building Multiple Sequence Alignment**). 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 names and the flag syntax of ClustalW. Yet, you will find that T-Coffee also has many extra possibilities. . .

5.2.1.6 Is T-Coffee very accurate?

T-Coffee belongs to the category of consistency based aligners which currently corresponds to the most accurate algorithms available (e.g. ProbCons, MSAprobs. . .). In addition, T-Coffee can combine (many) methods and therefore be as accurate (and hopefully more) as the methods it combines. For instance, the "accurate" mode of T-Coffee is very slow but also very accurate; on average this mode was shown to be 10 % more accurate than normal aligners on sequences less than 30% similar. If you need a very accurate alignment go to section **Building Multiple Sequence Alignment**.

5.2.2 What T-Coffee can and cannot do for you . . .

5.2.2.1 What T-Coffee can't do

To be honest, a short answer will be that there is only one thing T-Coffee cannot do for you: **T-Coffee can NOT fetch sequences for you**. You must select the sequences you want to align beforehand and prepare your own dataset. We suggest you use any BLAST server and format your sequences in FASTA so that T-Coffee can use them easily. The ExPASy BLAST server provides a nice interface for integrating database searches.

5.2.2.2 What T-Coffee can do

T-Coffee is not only just an aligner program, it comes with multiple tools and third party software increasing the range of its possibilities; here is a non exhaustive list of tasks T-Coffee can perform:

- **T-Coffee can compute (or at least try to compute!) accurate Multiple Sequence Alignments of DNA, RNA or Protein sequences**. Several modes and options are available and will be presented all along this manual. The default T-Coffee accepts any kind of sequence, although some modes are specific to a given type of sequence.

- **T-Coffee can help you to reformat, trim, clean, cut, color your input (sequences, structures...) or output (alignments, trees...) data**; meaning that once you have your data and/or results ready, you can always modify them at will.
- **T-Coffee allows you to combine results obtained with several alignment methods** (see the section **FAQ for T-Coffee** and **Building Multiple Sequence Alignment** for more details). T-Coffee can virtually combine all these MSAs you have to produce a new Multiple Sequence Alignment having the best agreement with all these methods you tried.
- **One of the most important improvement of T-Coffee is to let you combine sequences and structures**, so that your alignments are of higher quality. You need to have the SAP package installed to fully benefit of this facility (or to use another structural alignment method).
- **T-Coffee allows you to extract a collection of repeats from a single sequence or a set of sequences** using MOCCA. In other words, if you know the coordinates of one copy of a repeat, you can extract all the other occurrences. MOCCA needs some time to compute a library and then prompt you with an interactive menu. You just have to follow the instructions.
- **T-Coffee can be used to measure the reliability of your Multiple Sequence Alignment**. If you want to find out about that, read the section **FAQ for T-Coffee** or the **Technical Documentation** (-output flag). More details will be given anyway in this manual in the section **How Good Is Your Alignment?**.
- **T-Coffee can be used to compare alternative alignment**; in case you generate several alignments of the same sequences, you can compare these alignments using the most common scores (Sum-of-Pairs or Column Score). In case you have reference alignments, you can directly benchmark your method by comparing your MSAs to your references.

And probably many more options we will discover together all along this manual !

> **Warning:** Some options are carried out using the function "wget". If "wget" is not installed on your system, you can get it for free from wget download. Just type **wget** to be sure it is installed.

5.2.3 How does T-Coffee alignment works?

If you only want to make a standard Multiple Sequence Alignment, you may skip these explanations. But if you want to do more sophisticated things, these few indications may help before you start reading the documentation and the different articles.

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 christmas list: you can ask anything you fancy, it doesn't imply you will get it. Given a standard library, it is nearly impossible to have all the residues aligned at the same time because all the lines of the library may not agree. For instance:

```
Line 1 says:
Residue 1 of seq A with Residue 5 of seq B,
...
Line 100 says:
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, relatives, collaborators... and you will know exactly what we mean). You can generate this list of constraints the way you like. You may even provide it yourself, forcing important residues to be aligned by giving them high weights (see **FAQ for T-Coffee**). For your convenience, T-Coffee can generate (by 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.

Note: Be aware that nothing forces you to use a given library and that you could build it using other methods. In protein language, **T-Coffee is synonymous for freedom, the freedom of being aligned however you fancy** (I was probably a Tryptophan in some previous life).

5.3 Preparing Your Data

Important: About the syntax of T-Coffee command lines, just for you to know that it is quite flexible, for instance 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 completion provided by many shells, you can replace '=' and ',' with a space. Also, T-Coffee tools/modes are called using different flags to specify input/output files, parameters, modifiers, etc... Some flags are not always mandatory, however, if you use a correct/strict flag usage T-Coffee will always work fine; you just have some degrees of freedom ;-).

5.3.1 The reformatting utility: seq_reformat

5.3.1.1 General introduction

Nothing is more frustrating than downloading important data and realizing you need to format it before using it. In general, you should avoid manual reformatting: it is by essence inconsistent and will get you into trouble. It will also get you depressed when you realize that you have spent the whole day adding carriage return to each line in your files. T-Coffee comes with several tools to reformat/trim/clean/select your input data but also your output results, especially a very powerful reformatting utility named **seq_reformat**. You can use **seq_reformat** by invoking the t_coffee shell:

```
$$: t_coffee -other_pg seq_reformat
```

This will output the online flag usage of seq_reformat meaning a complete list of things seq_reformat can do for you. The seq_reformat is a reformatting utility so it recognizes automatically the most common formats (FASTA, Swiss-Prot,ClustalW, MSF, Phylip...). It reads the input file(s) via the **"-in"** and **"-in2"** flags and outputs in whatever specified format via the **"-output"** flag. In the meantime, you can use the flag **"-action"** to perform a wide range of modification on your data. In this section we give you quite a lot of different examples of you can do with **"-other_pg seq_reformat"**.

Danger: After the flag **-other_pg**, the common T-Coffee flags are not recognized anymore; it is like if you were using a different program.

Tip: When using T-Coffee seq_reformat, command line may become quite long...a practical way to handle this is to create in your .bashrc an alias to call directly seq_reformat. For example, write in your .bashrc: **alias reformat='t_coffee -other_pg seq_reformat'**. You can now call seq_reformat by tiping **reformat**.

5.3.1.2 Modification options

In order to perform different modifications on your data (residues/sequences/columns...), the **seq_reformat** utility can be followed by the flag **-action** and one or several modifiers listed here (this list is not exhaustive):

```
Options:
- +upper              : to uppercase your residues
- +lower              : to lowercase your residues
- +switchcase         : to selectively toggle the case of your residues
- +keep               : to only keep the residues within the range
- +use_cons +keep     : to only keep the columns within the range
- +remove             : to remove the residues within the range
- +convert            : to only convert the residues within the range
- +grep               : to select a given string of character
- +rm_gap             : to remove columns containing gaps
- etc...
```

5.3.1.3 Using a "cache" file

5.3.1.3.1 What is a cache in T-Coffee?

Several options can be performed easily by using what we call a cache (or cache file). In T-Coffee, a cache is a file containing an alternate version of your alignment where each position of the alignment is replaced by an alternative coding scheme. For instance each residue can be replaced by a score previously evaluated: this score can be the T-Coffee CORE index (cf. section **Evaluating Your Alignment**) or a matrix based evalution (blosum62mt or identity matrix). Then, when performing any modification or reformatting of your alignments, you can just specify the range of positions to be modified according to their respective scores within the cache. We will see some example especially regarding the modification of format of a given alignment; it is not mandatory to use a cache but it is quite practical. To generate a cache before any reformatting using a given evaluation score, you can use one of the following possible option:

```
Evaluating the T-Coffee CORE index during the alignment procedure:
$$: t_coffee sample_seq1.fasta -output=score_ascii

Evaluating the T-Coffee CORE index of a given alignment (-infile is mandatory):
$$: t_coffee -infile sample_seq1.aln -mode evaluate

Using an identity matrix:
$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -action +evaluate \
    idmat -output=score_ascii

Using a substitution matrix:
$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -action +evaluate \
    blosum62mt -output=score_ascii
```

5.3.1.3.2 Preparing a sequence/alignment cache

The following command will convert your alignment according to the given parameters: all A/a will be turned into 1, the gaps "-" will be conserved and all the other symbols (#) will be turned into 0. The flag **-action +convert** indicates the actions that must be carried out on the alignment before it is output into cache.

```
1) Initial alignment:
CLUSTAL FORMAT
```

(continues on next page)

(continued from previous page)

```
A CTCCGTgTCTAGGagt-TTACGTggAGT
B CTGAGA----AGCCGCCTGAGGTCG---
D CTTCGT----AGTCGT-TTAAGAca---
C -TTAAGGTCC---AGATTGCGGAGC---


2) Command line:
$$: t_coffee -other_pg seq_reformat -in=sample_dnaseq3.aln -output=clustalw_aln -
↪out=cache.aln \
    -action +convert 'Aa1' '.-' '#0'

3) Cache generated:
CLUSTAL W (1.83) multiple sequence alignment
A 0000000000100100-00100000100
B 000101----100000000100000---
D 000000----100000-00110101---
C -001100000---101000000100---
```

Other alternative are possible, for instance generating a cache for unaligned sequences (**-in** can refer to an alignment or unaligned sequences):

```
1) Command line:
$$: t_coffee -other_pg seq_reformat -in=sample_dnaseq3.aln -output=fasta_seq -
↪out=cache.seq \
    -action +convert 'Aa1' '.-' '#0'

2) Generate the file cache.seq:
>A
00000000001001000001000000100
>B
00010100001000000001000000000
>D
00000000001000000000110101000
>C
00011000000001010000000100000
```

5.3.1.3.3 Preparing a library cache

The library is a special format used by T-Coffee to declare relationships between pairs of residues. The cache library format can also be used to declare for instance the color of specific residues in an alignment. For instance, the following file `sample_dnaseq3.tc_lib` declares which residue of which sequence will receive which color. Note that the sequence number and the residue index are duplicated, owing to the recycling of this format from its original usage. It is also possible to use the BLOCK operator when defining the library (see **Technical Documentation**). The number right after BLOCK indicates the block length (10). The two next numbers (1 1) indicate the position of the first element in the block. The last value is the color.

```
! TC_LIB_FORMAT_01
4
A 27 CTCCGTgTCTAGGagtTTACGTggAGT
B 21 CTGAGAAGCCGCCTGAGGTCG
C 21 TTAAGGTCCAGATTGCGGAGC
D 20 CTTCGTAGTCGTTTAAGAca
#1 1
 +BLOCK+ 10 1 1 3
```

(continues on next page)

(continued from previous page)

```
 +BLOCK+ 5 15 15 5
#3 3
 6 6 1
 9 9 4
! CPU 240
! SEQ_1_TO_N
```

5.3.2 Modifying the format of your data

5.3.2.1 Keeping/Protecting your sequence names

Only few programs support long sequence names, and sometimes, when going through some pipeline the names of your sequences can be truncated or modified. To avoid this, **seq_reformat** contains a utility that can automatically rename your sequences into a form that will be machine-friendly, while making it easy to return to the human-friendly form.

1) **Create a code list**: The first thing to do is to generate a list of names that will be used in place of the long original name of the sequences:

```
$$: t_coffee -other_pg seq_reformat -in proteases_large.fasta -output \
    code_name > proteases_large.code_name
```

2) **Code your data**: This will create a file where each original name is associated with a coded name (Cxxx). You can then use this file to either code your dataset using the following command:

```
$$: t_coffee -other_pg seq_reformat -code proteases_large.code_name -in \
    proteases_large.fasta > proteases_large.coded.fasta
```

3) **Decode your data**: Then you can work with the file sproteases_large.coded.fasta and when you are done, you can decode the names of your sequences with the following command line:

```
$$: t_coffee -other_pg seq_reformat -decode proteases_large.code_name -in \
    proteases_large.coded.fasta
```

5.3.2.2 Changing the sequence format

Sometimes it may be necessary to change from one format to another, for instance when using another software which recognize only a given format. T-Coffee recognizes most common alignment formats and you can find the list of all input or output format recognized by simply typing **t_coffee -other_pg seq_reformat** without any input file(s). It is possible to reformat unaligned or aligned sequences alike although changing the alignment format is probably more interesting in order to use other applications; unaligned sequences format flags are generally preceded by the suffix **"_seq"** and aligned sequences flags by the suffix **"_aln"**. This also allows you to transform any alignment into unaligned sequences by removing the gaps. Here are some examples on how to change the format of your data:

```
For unaligned sequences (e.g. FASTA to PIR):
$$: t_coffee -other_pg seq_reformat -in proteases_small.fasta -output pir_seq > \
    proteases_small.pir_seq

For alignements (e.g. ClustalW to MSF):
$$: t_coffee -other_pg seq_reformat -in proteases_small.aln -output fasta_aln > \
    proteases_small.fasta_aln
```

(continues on next page)

(continued from previous page)

```
From aligned to unaligned sequences:
$$: t_coffee -other_pg seq_reformat -in proteases_small.aln -output fasta_seq > \
    proteases_small.fasta
```

Warning: Format recognition is not 100% full proof; occasionally you will have to inform the program about the nature of the file you are trying to reformat with **-input msf_aln -output fasta_aln** for instance.

5.3.2.3 Changing the case

5.3.2.3.1 Changing the case of your sequences

If you need to change the case of your sequences, you can use different modifiers embedded in **seq_reformat**. They are accessed via the **-action** flag. For instance, to write your sequences in lower case:

```
$$: t_coffee -other_pg seq_reformat -in proteases_small.aln -action +lower \
    -output clustalw
```

Tip: No prize for guessing that **+upper** will do exactly the opposite...

5.3.2.3.2 Changing the case of specific residues

If you want to change the case of a specific residue, you can use the flag: **+edit_residue <sequence> <residue #> <lower|upper|symbol>**. If you have more than one residue to modify, write all the coordinates in a text file (one coordinate per line) as spans are not yet supported; then give the file to T-Coffee

```
$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -action +upper \
    +edit_residue hmgb_chite 10 lower


$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -action +upper \
    +edit_residue sample_seq1.list
```

Warning: If you give a list of coordinates, it has to be a Unix text file (not a word document).

5.3.2.3.3 Changing the case with a cache

If you want to change the case depending on the score, you must either evaluate your alignment, or provide a cache. For example, this command line will upper the case of all residue then lower the case of every residue more than 50% identical to other residues in the same column:

```
Using a cache on the fly:
$$: t_coffee -other_pg seq_reformat -in sample_dnaseq2.aln -action +upper \
    +evaluate idmat +lower '[5-9]'

Using a cache file previously computed (2 steps):
```

(continues on next page)

(continued from previous page)

```
$$: t_coffee -other_pg seq_reformat -in sample_dnaseq2.aln -action +evaluate \
    idmat -output=score_ascii > sample_dnaseq2.cache

$$: t_coffee -other_pg seq_reformat -in sample_dnaseq2.aln -struc_in sample_dnaseq2.
↪cache \
   -struc_in_f number_aln -action +lower '[5-9]'
```

5.3.2.4 Coloring/Editing residues in an alignment

5.3.2.4.1 Changing the default colors

Colors are hard coded in the program, but if you wish, you can change them by simply creating a file named `seq_reformat.color` that is used to declare the color values. The name of the file (`seq_reformat.color`) is defined in `programmes_define.h`, COLOR_FILE and can be changed before compilation. By default, the file is searched in the current directory. For example, the following line written in `seq_reformat.color` indicates that the value 0 in the cache corresponds now to #FFAA00 in html, and in RGB 1, 0.2 and 0.

```
Format for hard coded colors in T-Coffee:
0 #FFAA00 1 0.2 0
```

5.3.2.4.2 Coloring specific types of residues/nucleic acids

You can color all the residues of your sequences on the fly; for instance, the following command line will color all the a's in color 0 (blue):

```
$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -action +3convert a0 \
    -output color_html > sample_seq1_color.html
```

> **Warning:** This option is case sensitive so the case of the residues or nucleotides should be the same in the command line (in this command line, only a lower case will be colored).

5.3.2.4.3 Coloring a specific residue of a specific sequence

If you want to color a specific residue/nucleotide, you can use the flag **+color_residue <sequence> <residue #> <color #>**. If you have more than one residue to color, you can put all the coordinates in a file, (one coordinate per line). Spans are not yet supported.

```
$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -action +color_residue \
    hmgb_chite 10 1 -output color_html > sample_seq1_single.html


$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -action +color_residue \
    sample_seq1_color.list -output color_html > sample_seq1_all.html
```

> **Warning:** If you give a list of coordinates, it has to be a Unix text file (not a word document).

5.3.2.4.4 Coloring according to the conservation

Use the +evaluate flag if you want to color your alignment according to its conservation level or using the boxshade scoring scheme:

```
Conservation color scheme:
$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -action +3evaluate pam250mt \
    -output color_html > color_cons.html

Boxshade color scheme:
$$: t_coffee -other_pg seq_reformat -in sample_aln1.aln -action +3evaluate boxshade \
    -output color_html > color_cons_box.html
```

5.3.2.4.5 Coloring an alignment using a cache

If you have a cache alignment or a cache library, you can use it to color your alignment and either make a post script, html or PDF output. For instance, if you use the file cache.seq:

```
 Produces a html file:
 $$: t_coffee -other_pg seq_reformat -in=sample_dnaseq3.aln -struc_in=sample_dnaseq3.
↪cache \
     -struc_in_f number_fasta -output=color_html -out=color_dnaseq3.html

Produces a pdf file:
 $*: t_coffee -other_pg seq_reformat -in=sample_dnaseq3.aln -struc_in=sample_dnaseq3.
↪cache \
     -struc_in_f number_fasta -output=color_pdf -out=color_dnaseq3.pdf

Produces an output using a library:
$$: t_coffee -other_pg seq_reformat -in=sample_dnaseq3.aln -struc_in=sample_dnaseq3.
↪tc_lib \
    -output=color_html -out=color_dnaseq3_lib.html
```

Warning: The script **ps2pdf** must be installed on your system for the pdf options.

5.3.3 Modifying the data itself...

5.3.3.1 Modifiying sequences in your dataset

5.3.3.1.1 Converting residues

It is possible for instance to selectively convert all given characters in a sequence (residues or nucleic acids alike) into another one, for example all G's having a score between 5 and 9 by using the command line:

```
$$: t_coffee -other_pg seq_reformat -in sample_dnaseq2.aln -struc_in sample_dnaseq2.
↪cache \
    -struc_in_f number_aln -action +convert '[5-9]' GX
```

5.3.3.1.2 Extracting sequences according to a pattern

You can extract any sequence by requesting a specific pattern to be found either in the name (NAME), the comment (COMMENT) or the sequence (SEQ) using the modifier is '+grep'. For instance, if you want to extract all the sequences whose name contain the word HUMAN, the flag NAME/COMMENT/SEQ indicates that the modification is made according to the sequences names, the comment section or the sequence itself, and the flag KEEP/REMOVE means that you will keep/remove all the sequences containing the string HUMAN. Here are some examples:

```
To keep sequences containing HUMAN in the name:
$$: t_coffee -other_pg seq_reformat -in proteases_large.aln -action +grep NAME \
    KEEP HUMAN -output clustalw

To remove sequences containing HUMAN in the name:
$$: t_coffee -other_pg seq_reformat -in proteases_large.aln -action +grep NAME \
    REMOVE HUMAN -output clustalw

To keep sequence which contain sapiens in the comment:
$$: t_coffee -other_pg seq_reformat -in proteases_large.fasta -action +grep COMMENT \
    KEEP sapiens -output clustalw

To remove sequences containing the pattern [ILM]K:
$$: t_coffee -other_pg seq_reformat -in proteases_large.aln -action +grep SEQ \
    REMOVE '[ILM]K' -output clustalw
```

Important: You should know that the pattern can be any perl legal regular expression, you can visit this page for some background on regular expressions.

Caution: This option is case sensitive (Human, HUMAN and hUman will not yield the same results). Be careful !!!

5.3.3.1.3 Extracting/Removing specific sequences by names

If you want to extract (command 1) or remove (command 2) several sequences in order to make a subset, you can specify a list of sequences by their full name:

```
Command 1: keep sequences
$$: t_coffee -other_pg seq_reformat -in proteases_small.fasta -action +extract_seq_
↪list \
    'sp|P29786|TRY3_AEDAE' 'sp|P35037|TRY3_ANOGA'

Command 2: remove sequences
$$: t_coffee -other_pg seq_reformat -in proteases_small.fasta -action +remove_seq \
    'sp|P29786|TRY3_AEDAE' 'sp|P35037|TRY3_ANOGA'
```

Note: Note the single quotes (') are mandatory as they are meant to protect the name of your sequence and prevent the Unix shell to interpret it like an instruction.

Once sequences are extracted or removed, some columns may remain containing only gaps, but it is possible to simply remove empty columns from the resulting dataset (command 3), and even extract specific blocks for the selected

sequences either keeping the exact same name (command 4) or the name of the specific blocks extracted (command 5):

```
Command 3: removing empty columns
$$: t_coffee -other_pg seq_reformat -in proteases_small.aln -action +extract_seq_list␣
↪\
    'sp|P29786|TRY3_AEDAE' 'sp|P35037|TRY3_ANOGA' +rm_gap

Command 4: keeping the initial name after extracting specific blocks and removing␣
↪empty columns
$$: t_coffee -other_pg seq_reformat -in proteases_small.aln -action +keep_name \
    +extract_seq 'sp|P29786|TRY3_AEDAE' 20 200 'sp|P35037|TRY3_ANOGA' 10 150 +rm_gap

Command 5: renaming sequences according to the extracted blocks and removing empty␣
↪columns
$$: t_coffee -other_pg seq_reformat -in proteases_small.aln -action +extract_seq \
    'sp|P29786|TRY3_AEDAE' 20 200 'sp|P35037|TRY3_ANOGA' 10 150 +rm_gap
```

Hint: The tag **+keep_name** must come BEFORE the tag **+extract_seq**.

5.3.3.1.4 Extracting the most informative sequences

Large datasets are problematic because they can be difficult to align and analyze, MSA programs tend to become very slow and inaccurate. In short, the best size for an MSA dataset would be between 20 to 40 sequences to have enough sequences to see the effect of evolution, but in the same time small enough so that you can visualize your alignment and recompute it as many times as needed. More important than its size, a good dataset have to be informative, when each sequence contains information the others do not have. The most informative sequences are the sequences that are as different as possible to one another, within your dataset. You can extract the most informative sequences using flag **+trim** followed by the number of sequences you wish to keep ("n" for a number and "N" for a perrcentage). The following commands will extract the 10 most informative sequences (command 1) or the 20% of most informative sequences (command 2):

```
Command 1:
$$: t_coffee -other_pg seq_reformat -in proteases_large.fasta -action +trim _seq_n10 \
    -output fasta_seq
Command 2:
$$: t_coffee -other_pg seq_reformat -in proteases_large.fasta -action +trim _seq_N20 \
    -output fasta_seq
```

Hint: The argument to trim include **_seq_**, it means your sequences are provided unaligned. If your sequences are already aligned, you do not need to provide this parameter. It is generally more accurate to use unaligned sequences.

Note: For very large dataset, **seq_reformat** will compute the similarity matrix between your sequences once only. It will then store it in its cache to be reused any time you run on the same dataset. In short this means that it will take much longer to run the first time, but be much faster if you need to rerun it.

5.3.3.1.5 Extracting/Removing sequences with the % identity

Removing too identical sequences (redundant)

Removing the most similar sequences is often what people have in mind when they talk about removing redundancy. You can do so using the **+trim** option. For instance, you can generate a dataset where no pair of sequences has more than 50% identity either from a dataset of unaligned sequences (command 1) or from any given alignment (command 2). If you start from unaligned sequences, the removal of redundancy can be slow. If your sequences have already been aligned using a fast method, you can take advantage of this by replacing the **"_seq_"** with **"_aln_"**. Just run the following command lines to see the difference un runtime:

```
Command 1: unaligned sequences
$$: t_coffee -other_pg seq_reformat -in proteases_large.fasta -action +trim _seq_%%50_

Command 2: aligned sequences
$$: t_coffee -other_pg seq_reformat -in proteases_large.fasta -action +trim _aln_%%50_
```

Note: Using aligned sequences results in a fastest trimming, however, it also means that you rely on a more approximate estimation of sequence similarity.

Removing too different sequences (outliers)

Sequences that are too distantly related from the rest of the set (called outliers) may have very negative effects on the overall alignment; to prevent this, it is advisable not to use them. The next command line will lead to the removal of all the sequences where no pair of sequences has less than 30% average accuracy with all the other sequences in the dataset (the symbol "_O" stands for Outliers) and more than 80% identity:

```
$$: t_coffee -other_pg seq_reformat -in proteases_large.fasta -action +trim _seq_%%80_
↪O30
```

Hint: This particular option is quite powerful as it allows you to decide both inferior and superior tresholds for trimming your dataset based on pairwise identity score, and therefore you can dissect your dataset according to different ranges of identity values. Be careful not to remove too many sequences ;-)

Forcing specific sequences to be kept

Sometimes you want to trim based on identity while making sure specific/important sequences remain in your dataset. You can do so by providing a pattern (**"_f"** for field) : it will keep all the sequences whose name contains the given string (**"_fNAME"**, **"_fCOMMENT"** or **"_fSEQ"**). Here are some examples corresponding to the different protected fields while removing all sequences above 50% identity:

```
Keep all HUMAN sequences
$$: t_coffee -other_pg seq_reformat -in proteases_large.fasta -action +trim \
    _seq_%%50_fNAME HUMAN

Keep all sequences containing ".apiens"
$$: t_coffee -other_pg seq_reformat -in proteases_large.fasta -action +trim \
    _seq_%%50_fCOMMENT '.apiens'

Keep all sequences containing residues
$$: t_coffee -other_pg seq_reformat -in proteases_large.fasta -action +trim \
    _seq_%%50_fSEQ '[MLV][RK]'
```

You can also specify the sequences you want to keep by giving another fasta file containing the name of these sequences via the flag **-in2**:

```
$$: t_coffee -other_pg seq_reformat -in proteases_large.fasta -in2 proteases_small.
↪fasta \
    -action +trim _seq_%%40
```

5.3.3.1.6 Chaining important sequences

In order to align two distantly related sequences, most multiple sequence alignment packages perform better when provided with many intermediate sequences that make it possible to 'bridge' your two sequences. The modifier **+chain** makes it possible to extract from a dataset a subset of intermediate sequences that chain the sequences you are interested in. For instance, let us consider the two sequences "sp|P21844|MCPT5_MOUSE" and "sp|P29786|TRY3_AEDAE" having 26% identity. This is high enough to make a case for a homology relationship between them, but this is too low to blindly trust any pairwise alignment. With the names of the two sequences written in the file sproteases_pair.fasta, run the following command:

```
$$: t_coffee -other_pg seq_reformat -in proteases_large.fasta -in2 proteases_pair.
↪fasta \
    -action +chain > proteases_chain.fasta
```

This will generate a dataset of 21 sequences, with the following chain of similarity between your two sequences. This is probably the best way to generate a high quality alignment of your two sequences when using a progressive method like ClustalW, T-Coffee, MUSCLE or MAFFT.

```
N: 21 Lower: 40 Sim: 25 DELTA: 15
#sp|P21844|MCPT5_MOUSE -->93 -->sp|P50339|MCPT3_RAT -->85 -->sp|P50341|MCPT2_M\
ERUN -->72 -->sp|P52195|MCPT1_PAPHA -->98 -->sp|P56435|MCPT1_MACFA -->97 -->sp\
|P23946|MCPT1_HUMAN -->81 -->sp|P21842|MCPT1_CANFA -->77 -->sp|P79204|MCPT2_SH\
EEP -->60 -->sp|P21812|MCPT4_MOUSE -->90 -->sp|P09650|MCPT1_RAT -->83 -->sp|P5\
0340|MCPT1_MERUN -->73 -->sp|P11034|MCPT1_MOUSE-->76 -->sp|P00770|MCPT2_RAT --\
>71 -->sp|P97592|MCPT4_RAT -->66 -->sp|Q00356|MCPTX_MOUSE -->97 -->sp|O35164|M\
CPT9_MOUSE -->61 -->sp|P15119|MCPT2_MOUSE -->50 -->sp|Q06606|GRZ2_RAT -->54 --\
>sp|P80931|MCT1A_SHEEP -->40 -->sp|Q90629|TRY3_CHICK -->41 -->sp|P29786|TRY3_A\
EDAE
```

5.3.3.2 Modifying columns/blocks in your dataset

5.3.3.2.1 Removing gapped columns

You can also remove all the columns containing a given proportion of gaps; for instance the following command will delete all the residues occurring in a column that contains 50% or more gaps (use 1 to delete residues from columns having 1 gap or more):

```
$$: t_coffee -other_pg seq_reformat -in sample_dnaseq3.aln -action +rm_gap 50
```

5.3.3.2.2 Extracting specific columns

Extracting portions of a dataset is something very frequently needed. You may need to extract all the sequences that contain the word human in their name, or you may want all the sequences containing a simple motif. We show you here how to do a couple of these things. To do this, you need an evaluation file that may have been generated with T-Coffee, either running a *de novo* alignment (command 1) or evaluating a preexisting alignment (command 2):

```
Command 1:
$$: t_coffee sample_seq1.fasta -output score_ascii, aln

Command 2:
$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -action +evaluate \
    blosum62mt -output score_ascii > sample_seq1_blosum62.score_ascii
```

This generates a score_ascii file that you can then use to filter out the bad bits in your alignment considering the individual score of each residue to trigger the filtering (command 3), or according to the whole column score by simply add the **+use_cons** flag (command 4). This last command can also be run on the fly with command 5. The commands 3/4/5 will keep only residues and/or columns having a score between 6 and 9.

```
Command 3:
$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -struc_in sample_seq1.score_
↪ascii \
    -struc_in_f number_aln -action +keep '[6-9]'

Command 4:
$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -struc_in sample_seq1.score_
↪ascii \
    -struc_in_f number_aln -action +use_cons +keep '[6-9]'

Command 5
$$: t_coffee -other_pg seq_reformat -in sample_aln1.aln -action +evaluate blosum62mt \
     +use_cons +keep '[6-9]'
```

Warning: Don't forget the simple quotes (‘), it's mandatory !!!

5.3.3.2.3 Extracting entire blocks

In case you want to extract a specific block of your alignment, for instance to remove poorly resolved regions, delimit your alignments boundaries or to extract domains, you can do so with the option **+extract_block**. In command 1, the option **cons** indicates that you are counting the positions according to the consensus of the alignment (i.e. the positions correspond to the columns # of the alignment). If you want to extract your block relatively to a specific sequence, you should replace cons with this sequence name (command 2).

```
Command 1: extract block from MSA
$$: t_coffee -other_pg seq_reformat -in proteases_small.aln -action +extract_block \
    cons 150 200

Command 2: extract_block relative to a give sequence of the MSA
$$: t_coffee -other_pg seq_reformat -in proteases_small.aln -action +extract_block \
    'sp|Q03238|GRAM_RAT' 10 200
```

Tip: It may be sometimes difficult to know where starts the blocks you are interested in except by counting manually the number of column. You can also make some tries by modifying the boundaries until you get the block you want and then redirect the result into the output file name of your choice.

5.3.3.2.4 Concatenating blocks or MSAs

If you have extracted several blocks generated using the previous command and you want to glue them together, you can use the **+cat_aln** modifier:

```
$$: t_coffee -other_pg seq_reformat -in proteases_small.aln -action +extract_block \
    cons 100 120 > block1.aln

$$: t_coffee -other_pg seq_reformat -in proteases_small.aln -action +extract_block \
    cons 150 200 > block2.aln

$$: t_coffee -other_pg seq_reformat -in block1.aln -in2 block2.aln -action +cat_aln
```

Note: The alignments do not need to have the same number of sequences and the sequences do not need to come in the same order.

5.3.4 Manipulating DNA sequences

5.3.4.1 Translating DNA sequences into protein sequences

If your sequences are DNA coding sequences, it is often safer and more accurate to align them as proteins (as protein sequences are more conserved than their corresponding DNA sequence). The **seq_reformat** options make it easy for you to translate your sequences:

```
$$: t_coffee -other_pg seq_reformat -in proteases_small_dna.fasta -action \
    +translate -output fasta_seq
```

5.3.4.2 Back-translation with the *bona fide* DNA sequences

Once your sequences have been aligned, you may want to turn your protein alignment back into a DNA alignment, either to do phylogeny, or maybe in order to design PCR probes. To do so, use the following command:

```
$$: t_coffee -other_pg seq_reformat -in proteases_small_dna.fasta -in2 \
    proteases_small.aln -action +thread_dna_on_prot_aln -output clustalw
```

5.3.4.3 Finding the *bona fide* sequences for the back-translation

Use the online server ProtoGen.

5.3.5 Manipulating RNA Sequences

5.3.5.1 Producing a Stockholm output: adding predicted secondary structures

5.3.5.1.1 Producing/Adding a consensus structure

Given an RNA multiple sequence alignment, it is possible to compute (command 1) or add (command 2) the alifold (Vienna package) consensus secondary structure and output in stockholm:

```
Command 1:
$$: t_coffee -other_pg seq_reformat -in sample_rnaseq2.aln -action +aln2alifold \
    -output stockholm_aln

Command 2:
$$: t_coffee -other_pg seq_reformat -in sample_rnaseq2.aln -action +add_alifold \
    -output stockholm_aln
```

5.3.5.1.2 Adding a precomputed consensus structure to an alignment

The file sample_rnaseq2.alifold contains the raw output of the alifold program produced via the RNAalifold webserver or captured with the command "RNAalifold <sample_rnaseq2.aln > sample_rnaseq2.alifold". It is possible to add this secondary structure to an alignment (command 1) and to stack Stockholm formatted secondary structures (command 2):

```
Command 1:
$$: t_coffee -other_pg seq_reformat -in sample_rnaseq2.aln -in2 sample_rnaseq2.
↪alifold \
    -input2 alifold -action +add_alifold -output stockholm_aln

Command 2:
$$: t_coffee -other_pg seq_reformat -in sample_rnaseq2.aln -in2 sample_rnaseq2.cons.
↪stk \
    -action +add_alifold -output stockholm_aln
```

Warning: The alifold structure and the alignment MUST be compatible. The function makes no attempt to thread or align the structure, it merely stacks it below the MSA.

5.3.5.2 Analyzing a RNAalifold secondary structure prediction

The following commands can either be applied on a Stockholm or a standard MSA. In the second case (standard MSA) the secondary structure will be automatically recomputed by alifold.

5.3.5.2.1 Visualizing compensatory mutations

The following command will output a color coded version of your alignment with matching columns indicated as follows:

- I: incompatible pair (i.e. at least one pair is not WC)
- N: pairs are Gus or WC
- W: all pairs are Watson
- c: compensatory mutations
- C: WC compensatory mutations

```
Standard alignment:
$$: t_coffee -other_pg seq_reformat -in sample_rnaseq2.aln -action +alifold2analyze␣
↪aln
```

(continues on next page)

(continued from previous page)

```
Color coded alignment:
$$: t_coffee -other_pg seq_reformat -in sample_rnaseq2.aln -action +alifold2analyze␣
↪color_html
```

Warning: Handling gapped columns: by default gapped column are ignored but they can be included by adding the tag **-usegap**.

5.3.5.2.2 Analyzing matching columns

The option **+alifold2analyze** will estimate the number of pairs of columns that are perfect Watson and Crick pairings, those that are neutral (including a GU) and those that include correlated mutations (command 1). The WCcomp are the compensated mutations maintaining WC base pairing. Other arguments can given, to display the list of paired positions and their status (compensated, Watson, etc...) use command 2:

```
Command 1:
$$: t_coffee -other_pg seq_reformat -in sample_rnaseq2.stk -action +alifold2analyze␣
↪stat

Command 2:
$$: t_coffee -other_pg seq_reformat -in sample_rnaseq2.stk -action +alifold2analyze␣
↪list
```

5.3.5.3 Comparing alternative folds

The folds associated with alternative alignments can be compared. This comparison involves counting how many identical pairs of residues are predicted on each sequence in one fold and in the other. The top of the output (@@lines) summarizes the results that are displayed on the input alignment; if the provided alignment do not have a fold, this fold will be estimated with alifold. The folds can be provided as Stockholm alignments:

```
$$: t_coffee -other_pg seq_reformat -in sample_rnaseq2.cw.stk -in2 sample_rnaseq2.
↪tcoffee.stk \
    -action +RNAfold_cmp
```

5.3.6 Phylogenetic Trees Manipulation

5.3.6.1 Producing phylogenetic trees

The seq_reformat is NOT a phylogeny package, yet over the time it has accumulated a few functions that make it possible to compute simple phylogenetic trees, or similar types of clustering. Given a multiple sequence alignment, it is possible to compute either a UPGMA or an NJ tree. The following commands use an identity matrix to compare your sequences and will output an unrooted NJ tree in Newick format (command 1) or a rooted UPGMA tree (command 2):

```
Command 1:
$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -action +aln2tree -output␣
↪newick \
    -out sample_seq1_tree_nj.nwk

Command 2:
```

(continues on next page)

(continued from previous page)

```
$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -action +aln2tree _TMODE_
↪upgma  \
    -output newick -out sample_seq1_tree_upgma.nwk
```

If your data is not data sequence, but a matrix of 1 and Os (i.e. SAR matrix for instance), you can use a different matrix to compute the pairwise distances (command 3), and all these parameters can be concatenated (command 4):

```
Command 3:
$#: t_coffee -other_pg seq_reformat -in sample_seq1.aln -action +aln2tree \
    _MATRIX_sarmat -output newick

Command 4:
$#: t_coffee -other_pg seq_reformat -in sample_seq1.aln -action +aln2tree \
    _TMODE_upgma_MATRIX_sarmat -output newick
```

Warning: Bootstrap facilities will also be added at some point... We recommend you to use Phylip or any other specific phylogenetic software (PhyML, RAxML, MrBayes, etc...) if you need some serious phylogeny !

5.3.6.2 Comparing two phylogenetic trees

A real interesting option is the ability to compare two trees (unrooted) returning some ofthe most common scores used for this including Robinson-Foulds

```
$$: t_coffee -other_pg seq_reformat -in sample_tree1.dnd -in2 sample_tree2.dnd -
↪action \
    +tree_cmp -output newick

#tree_cmp|T: 33 W: 20.00 L: 14.88 RF: 2 N: 9 S: 5
#tree_cmp_def|T: ratio of identical nodes
#tree_cmp_def|W: ratio of identical nodes weighted with the min Nseq below node
#tree_cmp_def|L: average branch length similarity
#tree_cmp_def|RF: Robinson and Foulds
#tree_cmp_def|N: number of Nodes in T1 [unrooted]
#tree_cmp_def|S: number of Sequences in T1
```

About the output scores in more details:

- T: Fraction of the branches conserved between the two trees. This is obtained by considering the split induced by each branch and by checking whether that split is found in both trees.
- W: Fraction of the branches conserved between the two trees. Each branch is weighted with MIN, the minimum number of leaf on its left or right.
- L: Fraction of branch length difference between the two considered trees.
- RF: is the standard Robinson-Foulds value when comparing trees.

The last line contains a tree where distances have been replaced by the number of leaf under the considered node:

- Positive values indicate a node common to both trees and correspond to MIN.
- Negative values indicate a node found in tree1 but not in tree2.
- The higher this value, the deeper the node.

Tip: You can extract this tree for further usage by typing **cat outfile | grep -v 'tree_cmp'**

5.3.6.3 Scanning phylogenetic trees

It is possible to scan an alignment and locally measure the similarity between an estimated local tree and some reference tree provided from an external source (or computed on the fly) using the following command:

```
$# : t_coffee -other_pg seq_reformat -in sample_seq1.aln -in2 sample_seq1_tree.nwk -
↪action \
     +tree_scan _MODE_scan__W_10_ > ph_tree_scan.txt
```

For each position of the alignment, W*2 blocks of size 2*1+1 up to W*2+1 will be extracted, for each of these block a tree will be estimated and the similarity of that tree with the reference tree will be estimated with cmp_tree. For each position, the tree giving the best fit will be reported, along with the size of the block leading to that tree:

```
P: <position> <block start> <block_end> <block score> <block Length>
```

5.3.6.4 Pruning phylogenetic trees

Pruning removes leaves from an existing tree and recomputes distances so that no information is lost. To do this with T-Coffee you need two input files: a tree file in the Newick format and a FASTA-like file where sequences can be omitted, but you can also leave them, at your entire convenience. The second file is merely a list of the sequences to be kept when pruning the tree. The resulting tree will contain only the sequences specified in the list.

```
Tree file: "sample_3Dseq1.tree"
((((TNFR10-2:0.04546,TNFR16-2:0.06640)...,TNFR4-2:0.05255) 45:0.00848); \

FASTA-like file: "sample_3Dseq1.fake"
>TNFR2-2
>TNFR4-4
...

Pruning the tree:
$$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.tree -in2 sample_3Dseq1.fasta -
↪action \
    +tree_prune -output newick
```

5.3.6.5 Tree Reformatting for MAFFT

The most common format for phylogenetic trees is newick, but some alternatives exist, including the one used by MAFFT for very large trees, in which the sequence names are replaced by their original index (1..N) in the sequence file. Seq_reformat can be used to cross convert these two formats:

```
$$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.tree -in2 sample_3Dseq1.fasta -
↪action +newick2mafftnewick
```

And the reverse operation when processing a mafft guide tree

:: $$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.mafftnewick -in2 sample_3Dseq1.fasta -action +mafft-newick2newick

5.3.7 Manipulating structure files (PDB)

5.3.7.1 Extracting a structure

There are many reasons why you may need a structure. T-Coffee contains a powerful utility named **extract_from_pdb** that makes it possible to fetch the PDB coordinates of a structure or its FASTA sequence without requiring a local installation. By default, the option **extract_from_pdb will** start looking for the structure in the current directory; it will then look it up locally (PDB_DIR) and eventually try to fetch it from the web (via a wget to the PDB). All these settings can be customized using environment variables (see next paragraph). For instance if you want to fetch the chain E of the PDB structure 1PPG and/or its sequence in FASTA format, you can use:

```
Fetch the structure:
$$: t_coffee -other_pg extract_from_pdb -infile 1PPGE

Fetch the correpsonding sequence:
$$: t_coffee -other_pg extract_from_pdb -infile 1PPGE -fasta
```

5.3.7.2 Adapting extract_from_pdb to your own environment

If you have the PDB installed locally, simply set the variable PDB_DIR to the absolute location of the directory in which the PDB is installed. The PDB can either be installed in its divided form or in its full form. If the file you are looking for is neither in the current directory nor in the local PDB version, extract_from_pdb will try to fetch it from rcsb. If you do not want this to happen, you should either set the environment variable NO_REMOTE_PDB_DIR to 1 or use the **-no_remote_pdb_dir** flag:

```
Setting up the environment:
##: export NO_REMOTE_PDB_FILE=1

Running PDB extract:
$$: t_coffee -other_pg extract_from_pdb -infile 1PPGE -fasta -no_remote_pdb_file
```

By default, T-Coffee also requires two important PDB files declared using the two following variables. These variables do not need to be set if the considered files are in the cache directory (default behavior):

```
##: export PDB_ENTRY_TYPE_FILE=<location of the file pdb_entry_type.txt>
(Found at: ftp://ftp.wwpdb.org/pub/pdb/derived_data/pdb_entry_type.txt)
and
##: export PDB_UNREALEASED_FILE=<location of the file unrealeased.xml>
(Found at: http://www.rcsb.org/pdb/rest/getUnreleased)
```

Warning: Since the file `unreleased.xml` is not part of the PDB distribution, T-Coffee will make an attempt to obtain it even when using the NO_REMOTE_PDB_DIR=1 mode. You must therefore make sure that the file PDB_UNREALEASED_FILE is pointing to is read and write.

5.4 Aligning Your Sequences

5.4.1 General comments on alignments and aligners

5.4.1.1 What is a good alignment?

This is a tricky question, a good answer would be **"a good alignment is an alignment that makes it possible to do good biology"**. In practice, the alignment community has become used to measuring the accuracy of alignment methods using structures. Structures are relatively easy to align correctly, even when the sequences have diverged quite a lot. The most common usage is therefore to compare structure based alignments with their sequence based counterpart and to evaluate the accuracy of the method using these criterions. Unfortunately it is not easy to establish structure based standards of truth. Several of these exist and they do not necessarily agree. To summarize, the situation is as roughly as follows:

- **Above 40% identity**, all the reference collections do agree with one another and all the established methods give roughly the same results. These alignments can be trusted blindly.
- **Below 40% identity**, all the reference collections stop agreeing and the methods do not give consistent results. In this area of similarity it is not necessarily easy to determine who is right and who is wrong, although most studies indicate that consistency based methods (T-Coffee, ProbCons, MAFFT-slow or MSAProbs) have an edge over traditional methods.

When dealing with distantly related sequences, the only way to produce reliable alignments is to use structural information. T-Coffee provides many facilities to do so in a seamless fashion. Several important factors need to be taken into account when selecting an alignment method:

- **The best methods are not always the best**. Given a difficult dataset, the best method is only more likely to deliver the best alignment, but there is no guaranty it will do so. It is very much like betting on the horse with the best odds.
- **The difference in accuracy between all the available methods is not incredibly high** (as measured on reference datasets). It is unclear whether this is an artifact caused by the use of 'easy' reference alignments, or whether this is a reality. The only thing that can change dramatically the accuracy of the alignment is the use of structural information.
- **Keep in mind that these methods have only been evaluated by comparison with reference alignments (benchmarks)**. This is merely one criterion among many. In theory, these methods should be evaluated for their ability to produce alignments that lead to accurate trees, good profiles or good models. Unfortunately, these evaluation procedures do not yet exist.

5.4.1.2 The main methods and their scope

Note: There are many MSA packages around, the most common ones being ClustalW, MUSCLE, MAFFT, T-Coffee and ProbCons; amongst the latest ones, you can find phylogeny aware aligners (PRANK and SATé) and modifed/improved consistency based aligners (MSAProbs). You can almost forget about the other packages, as there is virtually nothing you could do with them that you will not be able to do with these packages. All these packages offer a complex trade-off between speed, accuracy and versatility.

5.4.1.2.1 ClustalW is really everywhere...

ClustalW is still the most widely used Multiple Sequence Alignment package. Yet things are changing fast and different tests have consistently shown that ClustalW is neither the most accurate nor the fastest package around. This

being said, ClustalW is everywhere and if your sequences are similar enough, it should deliver a fairly reasonable alignment.

5.4.1.2.2 MAFFT/MUSCLE to align big datasets

If you have many sequences to align MUSCLE or MAFFT are the obvious choice. MAFFT is often described as the fastest and the most efficient. This is not entirely true, in its fast mode (FFT-NS-1), MAFFT is similar to MUSCLE and although it is fairly accurate, about 5 points less accurate than the consistency based packages (ProbCons and T-Coffee). In its most accurate mode (L-INS-i) MAFFT uses local alignments and consistency, however, it becomes much more accurate but also slower, and more sensitive to the number of sequences. More recently, we have seen growing the number of **(ultra) large scale** aligners such as Clustal Omega, PASTA, UPP, and we hope soon the large scale version of T-Coffee (called MEGA-Coffee).

Suitable for:

- Distance-based phylogenetic reconstruction (NJ trees)
- Secondary structure prediction

Not suitable for:

- Profile construction
- Structure modeling
- 3D prediction
- Function analysis

5.4.1.2.3 T-Coffee/ProbCons, slow but accurate !!!

T-Coffee works by first assembling a library and then by turning this library into an alignment. The library is a list of potential pairs of residues. All of them are not compatible and the job of the algorithm is to make sure that as many possible constraints as possible find their way into the final alignment: it is very much like trying to choose a meeting date, and each one says something like 'I need my Monday morning', 'I can't come on Thursday afternoon', and so on. In the end you want a schedule that makes everybody happy, if possible. The nice thing about the library is that it can be used as a media to combine as many methods as one wishes. It is just a matter of generating the right constraints with the right method and compile them into the library. ProbCons and MAFFT (L-INS-i) uses a similar algorithm, but with a Bayesian twist in the case of ProbCons. In practice, however, ProbCons and T-Coffee give very similar results and have similar running time. MAFFT is significantly faster.

Suited for:

- Profile reconstruction
- Structure modeling
- Function analysis
- 3D prediction

5.4.1.3 Choosing the right package (without flipping a coin !)

Each available package has something to go for it, it is just a matter of knowing what you want to do !! T-Coffee is probably the most versatile, but it comes at a price, its default aligner being currently slower than many alternative packages. In the rest of this tutorial we give some hints on how to carry out each of these applications within the T-Coffee framework.

Packages	MUSCLE	MAFFT	ProbCons	T-Coffee	ClustalW
Accuracy	++	+++	+++	+++	+
<100 Seq.	++	++	+++	+++	+
>100 Seq.	+++	+++	-	+	+
Remote Homologues	++	+++	+++	+++	+
MSA vs Seq.	-	-	+++	+++	+++
MSA vs MSA	-	-	-	+++	+++
>2 MSAs	-	-	-	+++	-
Seq. vs Struc.	-	-	-	+++	+
Splicing Var.	-	+++	-	+++	-
Reformat	-	-	-	+++	++
Phylogeny	-	-	-	+	++
Evaluation	-	-	+	+++	-
Speed	+++	+++	+	+	++

Table 1. Relative possibilities associated with the main packages. In any of the situations corresponding to each table line, (+++) indicates that the method is the best suited, (++) indicates that the method is not optimal but behaves reasonably well, (+) indicates that it is possible but not recommended (-) indicates that the option is not available.

Packages	MUSCLE	MAFFT	ProbCons	T-Coffee	ClustalW
Dist Based Phylogeny	+++	+++	++	++	++
ML or MP Phylogeny	++	+++	+++	+++	++
Profile Construction	++	+++	+++	+++	++
3D Modeling	++	++	++	+++	+
2D Predictions	+++	+++	++	++	++

Table 2. Most Suitable Appplications of each package. In any of the situations corresponding to each table line, (+++) indicates that the method is the best suited, (++) indicates that the method is not optimal but behaves reasonably well, (+) indicates that it is possible but not recommended (-) indicates that the option is not available.

5.4.2 Protein Sequences

5.4.2.1 General considerations

T-Coffee aligner is by default parallelized, meaning that it can use multiple cores when running on a cluster or a computer. By default, T-Coffee will use all available processors to run, but you can parallelize the differents steps and allocate the number of cores you want with the flag **-multi_core** or **n_core**. For more details, refer to the chapter **T-Coffee Technical Documentation**, subsection **CPU control**.

```
Using a single core:
$$: t_coffee -in sample_seq1.fasta -multi_core=msa

Using 12 cores:
$$: t_coffee -in sample_seq1.fasta -n_core=12
```

Also when running T-Coffee, it displays a lot of information directly on screening while running from general information, options, results, warnings...if you want, you can reduce the display using **-no_warning** to remove all the warnings, or even more strict using **-quiet** removing any display while running T-Coffee.

5.4.2.2 Computing a simple MSA (default T-Coffee)

T-Coffee default mode will simply compute a Multiple Sequence Alignment of the sequences you provided in input (command 1). It will display the final MSA on the screen and in several files according to the format you asked with command 2 (by default, the MSA is stored in a file .aln in ClustalW format). The headline of the alignment file contains important information such as the version of T-Coffee used, the CPU time, the overall consistency score (normalized to 100 or 1000 depending on the version of T-Coffee) and the total length of the MSA: it is quite practical to have a quick glance at the result.

```
Command 1: default MSA
$$: t_coffee proteases_small.fasta

Command 2: default MSA, multiple output files
$$: t_coffee proteases_small.fasta -output=clustalw,fasta_aln,msf
```

Each time you run T-Coffee, 3 files are always generated:

- `proteases_small.aln`: the alignment in ClustalW format
- `proteases_small.dnd`: the guide tree in Newick format
- `proteases_small.html`: the colored MSA in html format

Warning: The guide tree is not a phylogenetic tree, it is used in the alignment process for clustering the sequences.

Tip: You can visualize the colored html file with any browser/software you prefer. The display of the sequences should be aligned and formatted; if not, use another browser, it works quite well with Firefox, Safari, etc... If you need to do more sophisticated modifications on your MSA, we recommend to use Jalview which incorporate the T-Coffee color scheme.

5.4.2.3 Aligning multiple datasets/Combining multiple MSAs

If your sequences are spread across several datasets, you can give all the files you want (the limit is 200) via the flag **-seq**, and in any format you want. Just know that 1) if you give an alignment, the gaps will be reset and your alignment will only provide sequences, 2) sequences with the same name between two files are assumed to be the same sequence, 3) ff their sequences differ, they will be aligned and replaced by the consensus of that alignment (process known as sequence reconciliation). To align multiple datasets:

```
$$: t_coffee -seq=proteases1_small.fasta,proteases2_small.aln -output=clustalw,fasta_
↪aln,msf
```

You may also have a bunch of alignments (with the same sequences) that you have either precomputed, assembled manually or received from a colleague. You can also combine these alignments. For instance, let us imagine we generated 4 alignments with ClustalW using different gap penalties. To combine them into ONE single alignment, use the **-aln** flag. The final score indicates a high level of consistency (91%) between all these MSAs, meaning that the final MSA is probably correct.

```
Your 4 different MSAs:
##: clustalw -infile=proteases_small.fasta -gapopen=0 -outfile=g0.aln
##: clustalw -infile=proteases_small.fasta -gapopen=-5 -outfile=g5.aln
##: clustalw -infile=proteases_small.fasta -gapopen=-10 -outfile=g10.aln
##: clustalw -infile=proteases_small.fasta -gapopen=-15 -outfile=g15.aln
```

(continues on next page)

(continued from previous page)

```
Combining multiple MSAs:
$$: t_coffee proteases_small.fasta -aln g0.aln g5.aln g10.aln g15.aln -
↪output=clustalw,html
```

5.4.2.4 Estimating the diversity in your alignment

It is easy to measure the level of diversity within your MSA with the **-output** option of **seq_reformat**, it will output all the pairwise identities, as well as the average level of identity between each sequence and the others. There are two possibilities given that your input are unaligned sequences or not: **-output sim_idscore** realign your sequences pairwise so it can accept unaligned or aligned sequences alike; **-output sim** computes the identity using the sequences as they are in your input file so it is only suited for MSAs. You can after redirect, sort and grep the output in order to select the sequences you are interested in.

```
$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -output sim
```

5.4.2.5 Comparing alternative alignments

If you change the parameters, you will end up with alternative alignments. It can be interesting to compare them quantitatively. T-Coffee comes along with an alignment comparison module named **aln_compare**. You can use it to estimate the amount of difference between your two alignments either using the Sum-of-Pair score or the column score using the flag **-compare_mode** (sp or column). By default aln_compare returns the SoP score:

```
$$: t_coffee -other_pg aln_compare -al1 b80.aln -al2 b30.aln -compare_mode sp
```

This comparison will return the following result:

```
*****************************************************
seq1        seq2           Sim    [ALL]            Tot
b80            19          33.6     94.2 [100.0]    [79686]
```

The interpretation of this output is as follow: b80 is the reference MSA, it contains 19 sequences with an average identity of 33.6%, and is 94.2% identical to the second MSA b30.aln (79686 pairs to be precise). Of course, this does not tell you where are the good bits, but you can get this information for instance residues that have lost more than 50% of their pairing partner between the two alignments are in lower case (command 1) or converted in any character you want (command 2). The sp score is defined as the number of aligned residue pairs (i.e. not gaps) that are common between the reference (-al1) and the test (-al2) divided by the total numbers of pairs in the reference, while excluding gaps.

Other modes of comparison are available including the TC score

```
$$: t_coffee -other_pg aln_compare -al1 b80.aln -al2 b30.aln -compare_mode tc
```

The tc score is defined as the number of columns in the test MSA (-al2) that are common between the reference and the test, divided by the total number of columns in the reference. This measure, originally reported in Balibase can be problematic as it treats equaly highly and sparcely populated columns. For this reason the

```
$$: t_coffee -other_pg aln_compare -al1 b80.aln -al2 b30.aln -compare_mode column
```

The column score is defined as the total number pairs occuring in columns entirely identical between the reference (-al1) and the test (-al2) divided by the total number of pairs in the reference (excluding gaps). This makes the column score the equivalent of a weighted TC score.

It is also possible to print out the conserved positions between MSAs using the following commands.

```
Command 1:
$$: t_coffee -other_pg aln_compare -al1 b30.aln -al2 p350.aln -output_aln \
    -output_aln_threshold 50

Command 2:
$$: t_coffee -other_pg aln_compare -al1 b30.aln -al2 p350.aln -output_aln \
    -output_aln_threshold 50 -output_aln_modif x
```

Note: It is important to take into account that the comparison between -al1 and -al2 is not symetrical. It returns the fraction of -al1 residues pairs - by index - that are recovered in -al2. If -al1 is the reference alignment, this number is more or less the equivalent of a sensistivity measure (i.e. how much True positives are recovered). If -al1 is the tartget alignment, this measure becomes the equivalent of a specificity measure (i.e. what is the fraction of residue pairs in the target that are part of the reference).

Tip: aln_compare is particularly interesting if you are modifying the default parameters of T-Coffee and want to monitor the effects of your modifications.

5.4.2.6 Comparing subsets of alternative alignments

It is also possible to restrict the comparison to a specific subset of residues within the target alignment (for instance the core residues if using a structural reference, or the catalytic residues if using functional annotation alignment). This requires a cache in which residues are annotated as being part of the core or not. The cache is a plain text ascii file that can be produced in any way convenient to you. We show below how to produce a cache based on the consistency score and how to use this cache for further comparisons.

```
$$: t_coffee b11.fa -output aln score_ascii
```

This command generates two output files

- `b11.aln`: the alignment in ClustalW format
- `b11.score_ascii`: a local reliability score with every residue having a score between 0 and 9

The target file must then be reformatted so as to decide which residues will be part of the evaluation (c=> core) and which ones will be ignored. In this example we will set the threshild at 7 which means that the comparison will ignore all residues that do not have 70% or more support from the T-Coffee library:

```
$$: t_coffee -other_pg seq_reformat -in b11.score_ascii -input number_aln -action␣
↪+convert 0123456i 789c -output fasta_seq > b11.cache
```

One can then produce a target file with another method

```
$$: t_coffee b11.fa -method mafft_pair -outfile b11.tcmafft
```

With the cache, it is then possible to compare the original alignment (b11.aln) while only considering pairs of core residues

```
$$: t_coffee -other_pg aln_compare -al1 b11.aln -al2 b11.tcmafft -st b11.cache pep -
↪io_cat '[c][c]=[core]'
```

This comparison will return the following result:

:: seq1 seq2 Sim [core] Tot b11 11 32.8 99.5 [63.6] [26044]

The output can be interpreted as follows: the core residues - defined as c-c pairs constitute 63.6% of the -al1 alignment. 93.5% of these pairs are recovered in the -al2 alignment. Note that the -io_cat makes it possible to declare as many categories and combinations of categories as one wishes to have. For instance:

```
$$: t_coffee -other_pg aln_compare -al1 b11.aln -al2 b11.tcmafft -st b11.cache pep\
     -io_cat '[c][c]=[core];[c][i]=[mixed];[ci][ci]=[all1];[*][*]=[all2]'
```

5.4.2.7 Modifying the default parameters of T-Coffee

Note: The main parameters of T-Coffee are similar to those of ClustalW, including a substitution matrix and some gap penalties. In general, T-Coffee's default is adequate. If, however, you are not satisfied with the default parameters, we encourage you to change the following parameters. Interestingly, most of what we say here holds reasonably well for ClustalW.

5.4.2.7.1 Can you guess the optimal parameters?

Here is another tricky question. . . and the general answer is NO. The matrix and the gap penalties are simplistic attempts at modeling evolution. While the matrices do a reasonable job, the penalties are simply inappropriate: they should have a value that depends on the structure of the protein and a uniform value cannot be good enough. Yet, since we do not have better we must use them. . . In practice, this means that parameter optimality is a very *ad hoc* business. It will change from one dataset to the next and there is no simple way to predict which matrix and which penalty will do better. The problem is also that even after your alignment has been computed, it is not always easy to tell whether your new parameters have improved or degraded your MSA.

There is no systematic way to evaluate an MSA. In general, people visually evaluate the alignment, count the number of identical columns and consider that one more conserved column is good news. If you are lucky you may know a few functional features that you expect to see aligned. If you are very lucky, you will have one structure and you can check the gaps fall in the loops. If you are extremely lucky, you will have two structures and you can assess the quality of your MSA. An advantage of T-Coffee is the fact that the overall score of the alignment (i.e. the consistency with the library) is correlated with the overall accuracy. In other words, if you alignment score increases, its accuracy probably increases also. All this being said, consistency is merely an empirical way of estimating the change of parameters and it does not have the predictive power of a BLAST E-Value.

5.4.2.7.2 Changing the substitution matrix

T-Coffee only uses the substitution matrix to make the pairwise alignments that go into the library. These are all the global alignments of every possible pair of sequences, and the ten best local alignments associated with every pair of sequences.

- By default, these alignments are computed using a Blosum62 matrix, but you can use any matrix you fancy instead, including: pam120mt, pam160mt, pam250mt, pam350mt, blosum30mt, blosum40mt, blosum45mt, blosum50mt, blosum55mt, blosum62mt, blosum80mt, or even user-provided matrices in the BLAST format (see **T-Coffee Technical Documentation**).
- PAM matrices: These matrices are allegedly less accurate than the Blosum. The index is correlated to the evolutionary distances, you should therefore use the pam350mt to align very distantly related sequences.
- Blosum matrices: These matrices are allegedly the most accurate. The index is correlated to the maximum percent identity within the sequences used to estimate the matrix. you should therefore use the blosum30mt to align very distantly related sequences. Blosum matrices are biased toward protein core regions, explaining why

these matrices tend to give better alignments, since by design, they can capture the most evolutionary resilient signal contained in proteins.

Unless you have some structural information available, the only way to tell whether your alignment has improved or not is to look at the score. For instance, if you compute the two following alignments:

```
$$: t_coffee proteases_small.fasta -matrix=blosum30mt -outfile=b30.aln
$$: t_coffee proteases_small.fasta -matrix=blosum80mt -outfile=b80.aln
```

You will get two alignments that have roughly the same score but are slightly different. You can still use these two alternative alignments by comparing them to identify regions that have been aligned identically by the two matrices. These regions are usually more trustworthy.

5.4.2.7.3 Changing gap penalties

Important: Gap penalties are the core of the matter when it comes to MSAs. An interesting feature of T-Coffee is that it does not really need such penalties when assembling the MSA, because in theory the penalties have already been applied when computing the library. This is the theory, as in practice penalties can help improve the quality of the alignment.

The penalties can be changed via the flags **-gapopen** for the gap opening penalty and via **-gapext** for the gap extension penalty. The range for gapopen are [-500,-5000], the range for the extension should rather be [-1,-10]. These values do not refer to a substitution matrix, but rather to the values range of the consistency estimation (i.e. ratio) normalized to 10000 for a maximum consistency. The default values are **-gapopen=-50, -gapext=0**. The reasons for these very low values are that they are meant to be cosmetic only, since a trademark of T-Coffee (inherited from Dialign) is not to need explicit penalties. Yet, we know for a fact that alignments with higher gap penalties often look nicer (for publications) and are sometimes more accurate. For instance, you can try:

```
$$: t_coffee proteases_small.fasta -gapopen -100 -gapext -5
```

This gap penalty is only applied at the alignment level (i.e. after the library was computed). If you want to change the gap penalties of the methods used to build the library, you will need to go deeper... Two methods are used by default to build the library (command 1). One does global pairwise alignments and is named slow_pair, the other is named lalign_id_pair and produces local alignments. These methods are specified via the **-method** flag. Usually you do not need to write it because it is the default, but if you want to change the default parameters of the constituting methods (command 2), you will need to do so explicitly (the default parameters are for lalign_id_pair **GOP=-10, GEP=-4, MATRIX=blosum50mt** and for slow_pair **GOP=-10, GEP=-1 and MATRIX=blosum62mt**. Using the command 2, the library is now computed using the Blosum62mt with lalign, rather than the Blosum50mt; the good news is that when using this matrix, the score of our alignment increases from 48 to 50. We assume this new alignment is therefore more accurate than the previous one.

```
Command 1: default T-Coffee
$$: t_coffee proteases_small.fasta -method=lalign_id_pair,slow_pair

Command 2: modifiying the parameters
$$: t_coffee proteases_small.fasta -method lalign_id_pair@EP@MATRIX@blosum62mt, \
    slow_pair -outfile proteases_small.b62_aln
```

Warning: It only makes sense to compare the consistency score of alternative alignments when these alignments have been computed using the same methods (lalign_id_pair and slow_pair for instance).

5.4.2.7.4 Aligning (very) large datasets

The consistency based default method is not really suited for more that 200 sequences. Above this number, it is recommanded to use the regressive agorithm implemented in T-Coffee.

5.4.3 Protein sequences using 2D and/or 3D information

Using structural information when aligning sequences is very useful. The reason is that structures diverge slower than sequences. As a consequence, one may still find a discernable homology between two sequences that have been diverging for a long time beyond recognition using their corresponding structure. Yet, when assembling a structure based MSA, you will realize that these sequences contain key conserved residues that a simple alignment procedure was unable to reveal. We show you in this section how to make the best of T-Coffee tools to incorporate structural information in your alignment.

5.4.3.1 Using 3D structures: Expresso/3D-Coffee

5.4.3.1.1 Requirements to run Expresso/3D-Coffee

Expresso needs BLAST you provide with the tag **-blast** and it also requires the PDB database you can specify via the flag **-pdb_db**. The good news is that it is not mandatory to have BLAST or the PDB installed locally as T-Coffee can automatically fetch the structures directly from RCSB (the home of PDB) using EBI BLAST web service.

5.4.3.1.2 How does Expresso/3D-Coffee work?

Expresso/3D-Coffee is probably one of the most important improvement of T-Coffee. The principle is simple: 1) it first runs a BLAST for every sequence in your dataset against the PDB and finds (or not) a structure similar in sequence (35% identity by default) to be used as a template for structurally aligning your sequence, 2) the correspondence between the query sequences and the templates are stored in a template file which is automatically generated by Expresso. Here lies the difference between 3D-Coffee and Expresso: when running Expresso, fetching structures and creating the template file are automated (so you can reuse it for other applications) while using 3D-Coffee is a bit more tricky as it requires the name of the sequences to correspond to the structure file name (and it does not fetch or create anything for you). Of course, you can create and use your own template with the tag **-template_file** with the same format presented here. At the end, whenever there are enough templates (minimum of two obviously), it will align sequences using the structural information, otherwise sequences will be aligned using the standard T-Coffee aligner. Of course, if your dataset only contains structures, your alignment becomes a structural alignment.

```
>sp|P08246|ELNE_HUMAN  _P_ 1PPGE
>sp|P20160|CAP7_HUMAN  _P_ 1AE5
...
```

Tip: The PDB files used as a template should be in the current directory, otherwise you have to declare in the template file the whole path to find your templates.

5.4.3.1.3 Running Expresso/3D-Coffee

Both Expresso (command 1) and 3D-Coffee (command 2) are modes of T-Coffee you can call with the flag **-mode**; the correspond to a preestablished configuration with default parameters. The template selection is indicated with the flag **-template_file** followed by either **PDB** (means BLAST will run remotely on the PDB) or **_SELF_P_** (means that

the PDB identifier is the name of the sequence so there is no need to run BLAST) or a template file of your choice. As you can see in commands 1 and2, SAP (sap_pair) is the default structural aligner but you can choose alternative aligner(s) installed. You can give any combination of methods with the flag **-method**, but at least one has to be a structural aligner (command 3). You can also specify local installations of BLAST and PDB (command 4) **which is highly recommended if you want reproducible results**.

```
Command 1: two ways of running Expresso
$$: t_coffee three_pdb_two_seq.fasta -mode expresso
$$: t_coffee three_pdb_two_seq.fasta -method sap_pair,slow_pair -template_file PDB

Command 2: two ways of running 3D-Coffee
$$: t_coffee three_pdb.fasta -mode 3dcoffee
$$: t_coffee three_pdb.fasta -method sap_pair,slow_pair -template_file _SELF_P_

Command 3: Choosing your own templates and methods
$$: t_coffee three_pdb_two_seq.fasta -method mustang_pair,slow_pair -template_file \
    three_pdb_two_seq_pdb1.template_list

Command 4: Running Expresso using a local BLAST/PDB
##: t_coffee three_pdb_two_seq.fasta -mode expresso -blast=LOCAL -pdb_db=<PDB> \
    -pdb_type d -pdb_min_sim 95 -pdb_min cov 90 -cache $PWD
```

Tip: By default, structures and structural pairwise alignments are stored in your local ~/.t_coffee/cache/ allowing Expresso to run faster if you reuse similar structures; you can choose to have all these files directly in your working directory by using **-cache=$PWD**. Don't forget to empty your cache directory from time to time otherwise your folder is just getting bigger and bigger (similar comment can be done for any template based mode of T-Coffee).

5.4.3.1.4 Template search paramaters

To use Expresso, you have different option from an entirely automated procedure to tailored procedure, by selecting either your own structures or by defining different criteria for the template selection. You can have an exhaustive list in the **T-Coffee Technical Documentation** (subsection **Template based T-Coffee modes**) yet the most important parameters for the template selection are the following:

- **-pdb_type** : type of structure ("d" for diffraction/XRAY or "n" NMR structures or "e" EM structures)
- **-pdb_min_cov** : minimum coverage between query sequence and template (from 0-100%)
- **-pdb_min_sim** : minimum identity between query sequence and template (from 0-100%)

5.4.3.2 Aligning sequences and structures

5.4.3.2.1 Mixing sequence profile and structure templates

If you want to go further, and be even slower, you can use the accurate mode that will combine profile and structural information. If no structure is available, the template will be a profile (similar to PSI-Coffee, see subsection **Aligning Profiles**). It is probably one of the most accurate way of aligning sequences currently available as it tries to get as much information as possible.

```
$$: t_coffee sample_seq1.fasta -mode accurate
```

5.4.3.2.2 Aligning profile using structural information

If you have two profiles to align, an ideal situation is when your profiles each contain one or more structures. These structures will guide the alignment of the profiles, even if they contain very distally related sequences. All you need is a template file that declares which sequences have a known structure. If you only want to align sequences, you can try:

```
$$: t_coffee -profile=profile1_pdb1.aln, profile2_pdb2.aln -method sap_pair \
    -profile_template_file two_profiles.template_file
```

5.4.3.3 Using secondary structure predictions

T-Coffee can be used to predict secondary structures and transmembrane domains. For secondary structure predictions, the current implementation is only able to run GOR on either single sequences or on a bunch of homologues found by BLAST.

5.4.3.3.1 Single sequence prediction

To make a secondary structure prediction T-Coffee used the GOR software. In the command line **-template_file=SSP** is a hard coded mode which prompts the computation of predicted secondary structures. Used this way, the method will produce for each sequence a secondary prediction file (`<sequence_name>.ssp`). GOR run on single sequences with a relatively low accuracy, which can be increased by coupling it with BLAST (command 2). When doing so, the predictions for each sequence are obtained by averaging the GOR predictions on every homologue as reported by a BLAST against nr. By default the BLAST is done remotely at the NCBI using the BLASTPGP web service of the EBI. Transmembrane structures can also be carried out simply, or following the same previous strategy (command 3 and 4).

```
Command 1:
$#: t_coffee sample_seq1.fasta -template_file SSP

Command 2:
$#: t_coffee sample_seq1.fasta -template_file PSISSP

Command 3:
$$: t_coffee sample_seq1.fasta -template_file TM

Command 4:
$$: t_coffee sample_seq1.fasta -template_file PSITM
```

5.4.3.3.2 Incorporation of the prediction in the alignment

It is possible to use the secondary prediction (command 1) or the transmembrane domains prediction (command 2) in order to reward the alignment of similar elements:

```
Command 1:
$#: t_coffee sample_seq1.fasta -template_file PSISSP -method_evaluate_mode ssp -
↪method \
    lalign_id_pair

Command 2:
$$: t_coffee sample_seq1.fasta -template_file PSITM -method_evaluate_mode tm -method \
    lalign_id_pair
```

The overall effect is very crude and can go up to overweighting by 30% the score obtained when matching two residues in a similar secondary structure state. The net consequence is that residues in similar predicted states tend to be aligned more easily.

Hint: In that case, the flag **-method** can only accept one single method

5.4.3.3.3 Using other secondary structure predictions

If you have your own predictions, you can use them to run T-Coffee providing you give your own template file, where the file containing the secondary structure prediction is declared along with the sequence (with _E_).

```
Command:
$#: t_coffee sample_seq1.fasta -template_file sample_seq1_ssp.template -method_
↪evaluate_mode \
    ssp -method lalign_id_pair

Format of the template file:
>hmgl_wheat _E_ hmgl_wheat.ssp
>hmgb_chite _E_ hmgb_chite.ssp
>hmgl_trybr3 _E_ hmgl_trybr3.ssp
...

Format of the prediction file:
>hmgl_wheat
CCCCCCCCCCCCCHHHHHHHCCCCCCCCCCHHHHHHHHHHHHHHHHCCCCHHHHHHHHHHHHHHHCE
```

5.4.3.3.4 Output of the prediction

You can output a color coded version of your alignment using the secondary predicted structure or transmembrane regions predictions:

```
Secondary structure prediction:
$#: t_coffee sample_seq1.fasta -template_file PSISSP -output sec_html

Transmembrane regions prediction:
$$: t_coffee sample_seq1.fasta -template_file PSITM -output tm_html
```

5.4.4 RNA sequences using 2D and 3D Structure

The main property of RNA sequences is to have a secondary structure that can be used to guide the alignment. While the default T-Coffee has no special RNA alignment method incorporated in, we have developped specific modes and tools for RNA alignment and analysis (see subsection **Manipulating RNA Sequences** for more details).

5.4.4.1 RNA sequences using 2D structure (R-Coffee)

5.4.4.1.1 Introduction

R-Coffee is the special mode of T-Coffee developped to handle specifically RNA sequences. It has been proven far more accurate than T-Coffee default because of its specific design. It can be run as a standalone aligner (using secondary structure prediction) or using third party software.

5.4.4.1.2 R-Coffee: aligning RNA sequences using secondary structures

R-Coffee uses predicted secondary structures via the software RNApfold, in order to improve RNA alignments. Running R-Coffee by default is rather simple (command 1) but as for T-Coffee, you can also specify the methods you prefer (command 2):

```
Command 1: R-Coffee default
$$: t_coffee sample_rnaseq1.fasta -mode rcoffee

Command 2: R-Coffee selected methods
$$: t_coffee sample_rnaseq1.fasta -mode rcoffee -method lalign_id_pair,slow_pair
```

5.4.4.1.3 Improving R-Coffee

There are two modes we proposed to improve R-Coffee alignments: 1) using the best method for RNA sequences (namely Consan), 2) combining many methods to achieve a better reliability (RM-Coffee).

```
Using Consan (best):
$#: t_coffee sample_rnaseq1.fasta -mode rcoffee_consan


Using multiple methods:
$$: t_coffee sample_rnaseq1.fasta -mode rmcoffee
```

Tip: In order to know if a RNA alignment is better than another one, the best is to visualize the compensatory mutations of the secondary structure: look at the subsection **Manipulating RNA Sequences**.

5.4.4.2 RNA Sequences using 2D and 3D structures (SARA-Coffee and RSAP-Coffee)

SARA-Coffee is a structure based multiple RNA aligner. This is a new algorithm that joins the pairwise RNA structure alignments performed by SARA with the multiple sequence T-Coffee framework. Since setting up the SARA-Coffee dependencies (T-Coffee, SARA, X3DNA, Numpy, Biopython, Perl, Python 2.7) can be tricky we provide a self-contained Vagrant VM, which downloads and configures all the required pieces of software for you. This procedure can be complex, and if you cannot install SARA, we recommand using RSAP-Coffee instead (cf next paragraph).

RSAP-Coffee is a an an alternative experimental mode of SARA-Coffee using protein structral aligners. It processes RNA PDB files so that the C3' Carbons are treated as C-alpha in a protein chain. This procedure is suitable for any protein structural aligner that only considers alpha carbons in the main chain. By default, rsapcoffee uses the sap_aligner that ships with T-Coffee default distribution.

Note: SARA-Coffee and RSAP-Coffee use very similar command line. In practice, the only difference is the use of the method sa_pair instead of the sara_pair methiod when buolding the T-Coffee library.

5.4.4.2.1 Running SARA/RSAP-Coffee

The procedure is similar to Expresso/3Dcoffee and requires providing a template file in the PDB files are matched with the corresponding sequences

```
><RNA Seq Identifier>  _P_ <PDB>
><RNA Seq Identifier>  _P_ <PDB>
...
```

Note: PBD can be either a file PDB identifier or a PDB file. The file can be either in your homew dir or on a location specified by the environement variable **PBD_DIR**. If you use an identifier T-Coffee will be able to download the file from RCSB.

Note: If in your PDB the SEQRES and the ATOM yield a different sequence, T-Coffee will use a sequence estimated from ATOM. If tghis sequence does not match your FASTA, they will be reconciliated using an alignment procedure to establish residues equivalencies.

You then need to run

```
$$: t_coffee t_coffee -seq rna3D.fa -mode rsapcoffee -template_file rna3D.template
```

Or $$: t_coffee t_coffee -seq rna3D.fa -mode saracoffee -template_file rna3D.template

By Default, RSAP-Coffee only runs sap, but you can also run other structural methods supported by T-Cofee. like Mustang or TM-Align

```
$#: t_coffee <yourseq.fasta> -mode rsapcoffee -method sap_pair, mustang_pair, TMalign_
↪pair -template_file <template file>
```

The default procedure is meant to handle a muxture of PDB files and sequences. All possble PDB sequences pairs are aligned using strucure based methods (structural library) and all possible sequence pairs (including PDB sequences) sequences are then aligned using the R-Coffee mode (secondary library). In the secondary library, the PDB sequences are aligned using their experimentallyerived secondary structures, as estimated by X3DNA. Note that id you aonly have structures, you may use the folloing command lines to carry out a strict structure based sequence alignment:

```
$$: t_coffee -seq rna3D.fa -method sap_pair -template_file rna3D.template
```

In that case the PDB Sequences are aligned using the experimental

5.4.4.2.2 Estimating the accuracy of an RNA structure based alignment

It is possible to use the irmsd procedure to estimate d-RMSD of your sutcure based sequence alignmnent

```
$$: t_coffee -other_pg irmsd -aln rna3D_msa2.aln -template_file rna3D.template
```

5.4.4.2.3 Installing SARA-Coffee VM

Follow the procedure:

```
1) Install or update virtual box from https://www.virtualbox.org/wiki/Downloads

2) Install or update vagrant from: http://www.vagrantup.com

3) Clone the sara-coffee virtual machine (this project) in a convenient location
   ##: git clone https://github.com/cbcrg/sara-coffee-vm.git
```

(continues on next page)

(continued from previous page)

```
4) Enter the sara-coffee-vm folder and launch vagrant
   ##: cd sara-coffee-vm/
   ##: vagrant up
```

The first time you run it, it will automatically download the virtual machine and all the packages required by SARA-Coffee. It may take some minutes to complete, so be patient. When it boots up and the configuration steps are terminated, login into the VM instance:

```
1) Login in VM:
   ##: vagrant ssh

2) Go to SARA-Coffee:
   ##: cd sara_coffee_package/

3) Run SARA-Coffee:
   ##: ./sara_coffee.sh <input file> <output file>
```

The folder '/vagrant/' is shared between the Sara-Coffee virtual and your local machine. On your local machine, this folder is the one in which you started vagrant (i.e. sara-coffee-vm). When finished, stop the VM using the command **vagrant halt** or **vagrant destroy**, depending if you want to temporary stop the execution or delete permanently the VM with all its files.

5.4.4.2.4 Docker image for SARA-Coffee

SARA-Coffee is also distributed as a Docker container. This will allow you to run it without having to install and configure each single dependency packages. If you have Docker installed simply pull the SARA-Coffee container by using the command 1 and run SARA-Coffee using the command 2:

```
Command 1: Pull SARA-Coffee container
##: docker pull cbcrg/saracoffee**.

Command 2: Run SARA-Coffee
##: docker run -v $PWD:$PWD -w $PWD cbcrg/saracoffee <input> <output>**
```

Note: this command assumes your input file is located in the working directory. If this is not the case, you will need to mount the input file path accordingly.

5.4.5 Aligning DNA sequences

5.4.5.1 Aligning DNA sequences

MSA methods are not at their best when aligning DNA sequences. Whenever you can, try using a local MSA package like the Gibbs sampler; yet if you believe your DNA sequence are homologous over their entire length, you can use T-Coffee. In theory, the program automatically recognizes DNA sequences and uses appropriate methods, yet adding the **-type=dna flag** cannot do any harm...

```
$$: t_coffee sample_dnaseq1.fasta -type=dna
```

The type declaration (or its automatic detection) triggers the use of the appropriate substitution matrix in most of the methods. However, if you would rather use your own matrix, use the following command where you should replace **idmat** with your own matrix in BLAST format.

```
$$: t_coffee sample_dnaseq1.fasta -in Mcdna_fast_pair@EP@MATRIX@idmat
```

5.4.5.2 Pro-Coffee: Aligning functional DNA regions

Pro-Coffee is a MSA method specifically designed for promoter regions or other orthologous DNA regions containing functional elements (enchancers for instance). Pro-Coffee takes nearest-neighbour nucleotide correlations into account when aligning DNA sequence. For this it first translates sequences into a dinucleotide alphabet and then does the alignment using a specifically designed dinucleotide substitution matrix. This matrix was constructed from binding sites alignments from the Transfac database. A benchmark on multispecies ChIP-seq data shows that validated binding sites will be better aligned than when using off-the-shelf methods. To run Pro-Coffee is easy by defaults (command 1), however the user can change empirically all gap parameters in order to improve the MSA (command 2). By default, parameters were optimized on benchmarks (GOP=-60/GEP=-1) but as every dataset is different you may want to see for yourself. Once your alignment is done, you can always use reformatting options in order to extract the regions of interest (command 3).

```
Command 1: Pro-Coffee default
$$: t_coffee -seq c18orf19.fa -mode procoffee

Command 2: Pro-Coffee parameters
$$: t_coffee -seq c18orf19.fa -method promo_pair@EP@GOP@-60@GEP@-1

Command 3: Extracting regions
$$: t_coffee -other_pg seq_reformat -in c18orf19.aln -action +extract_block \
    'ENSG00000177150' 1852 1983 > c18orf19_chipseq.aln
```

Tip: If you want more details, we suggest you follow the subsection **Quick Start, Tutorial (Practical Examples)** published in Nature Protocols (2011) or refer to the original article.

5.4.5.3 Splice variants

Splice variants are especially challenging for most MSA programs, because the splicing variants need very long gaps to be inserted, and most programs attempt to match as many symbols as possible. Standard programs like ClustalW or MUSCLE are not good at dealing with this situation and in our experience, the only programs that can do something with splice variants are those using local information like some flavors of MAFFT and T-Coffee. For instance, if you try muscle on the following dataset with the command 1, you will quickly realize your alignment is not very good and does not show where the alternative splicing co-occurs. On the other hand, if you use T-Coffee (command 2), things become much clearer. The reason why T-Coffee does better than other packages is mostly because it uses local information (lalign_id_pair) and is therefore less sensitive to long gaps. If the default mode does not work for your dataset, you can try to be a bit more aggressive and only use local information to compute your library (command 3). Of course, the most distantly related your sequences, the harder the alignment of splice variants.

```
Command 1: using MUSCLE
##: muscle -in sv.fasta -clw -out sv_muscle.aln

Command 2: using T-Coffee
$$: t_coffee sv.fasta
```

(continues on next page)

(continued from previous page)

```
Command 3: using only local information
$$: t_coffee sv.fasta -method lalign_id_pair
```

5.4.5.4 Noisy coding DNA sequences...

When dealing with coding DNA, the right thing to do is to translate your DNA sequence and thread the DNA onto the protein alignment if you really need some DNA. However, sometimes, your cDNA may not be so clean that you can easily translate it (frameshifts and so on). Whenever this happens, try (no warranty) the following special method. The test case in three_dna_seq.fasta contains the DNA sequences of three proteases with a couple of frameshifts here and there. If you make a regular alignment of these sequences (command 1) you see that many gaps have sizes that are not multiple of 3 (codon size). When using an appropriate alignment method that takes into account all the frames at the same time, we get something much more meaningful (command 2). At the end, you can even recover the corrected protein sequence (command 3) using a special option **+clean cdna**, a small HMM that loops through each sequence and select the frame in order to maximize the similarity within the alignment (part of the **seq_reformat** utility).

```
Command 1: Default alignment
$$: t_coffee three_cdna.fasta

Command 2: Special mode for cDNA
$$: t_coffee three_cdna.fasta -method cdna_fast_pair

Command 3: Recovering your protein sequences
$$: t_coffee -other_pg seq_reformat -in three_cdna.aln -action +clean_cdna +translate
```

5.4.6 Using many MSA methods at once

One of the most common situation when building MSAs is to have several alignments produced by different alternative methods, and not knowing which one to choose. In this section, we show you how to use M-Coffee to combine many alignments into one single alignment, or how you can specify only the methods you want. M-Coffee is not always the best method, but extensive benchmarks on BAliBASE, PREFAB and HOMSTRAD have shown that it delivers the best alignment 2 times out of 3. If you do not want to use the methods provided by M-Coffee, you can also combine precomputed alignments.

5.4.6.1 Using third party aligner via T-Coffee

T-Coffee is installed along with many aligners necessary to run M-Coffee for instance, and many more. If you type **t_coffee**, it will display on the screen the different t_coffee options and all the methods included. If you look carefully, you will see that most of the methods exist under two denominations: 1) **<method>_msa** or 2) **<method>_pair**. In the first case, it means that T-Coffee will use the specified method to run your MSA, so you can easily have a ClustalW or a MAFFT alignment using T-Coffee. In the second case, you ask T-Coffee to align every pair of sequence with the specified methods, the final MSA will be computed using the T-Coffee consistency between all the pairs. Go to the **Integrating External Methods in T-Coffee** if you want more information.

5.4.6.2 Using all the methods at the same time: M-Coffee

To use M-Coffee (M stands for Meta aligner), you will need several packages to be installed (see **T-Coffee Installation** and section **Integrating External Methods in T-Coffee**). If you did a default installation, all the software you need should be there. M-Coffee is a special mode of T-Coffee that you can call using the flag **-mode mcoffee**. It will align your sequence using 8 different aligners: ClustalW, POA, MUSCLE, ProbCons, MAFFT, Dialing-T, PCMA and T-Coffee:

```
$$: t_coffee proteases_small.fasta -mode mcoffee -output clustalw, html
```

The final MSA is a combination of all methods. The alignment is colored with the T-Coffee consistency color scheme, but in this case the colors will reflect the consistency between methods: 1) regions in red have a high consistency, so all the methods agree and you can expect them to be fairly accurate, 2) regions in green/blue have the lowest consistency, meaning that all the methods deliver different alignment in these regions and you should not trust them. Overall this alignment has a score of 951 (1000 being the max), which means that it is roughly 95% consistent with the entire collection; this is a fairly high index meaning that you can trust your alignment.

5.4.6.3 Using selected methods to compute your MSA

Using the 8 methods predefined in M-Coffee can sometimes be a bit heavy, if you only want to use a subset of your favorite methods, you should know that each of these methods is available via the **-method** flag. You can make all the combination you want !!! For instance, to combine MAFFT, MUSCLE, T-Coffee and ProbCons, you can use:

```
$$: t_coffee proteases_small.fasta -method=t_coffee_msa,mafft_msa,probcons_msa, \
    muscle_msa -output=html
```

5.4.7 Aligning profiles

Sometimes, it is better to prealign a subset of your sequences, and then to use this small alignment as a master for adding sequences (sequence to profile alignment) or even to align several profiles together if your protein family contains distantly related groups. T-Coffee contains most of the facilities available in ClustalW to deal with profiles, and the strategy we outline here can be used to deal with large datasets.

5.4.7.1 Aligning sequence(s) to profile(s)

Assuming you have multiple alignment(s) (sproteases_small.aln) or profile(s) here is a simple strategy to align sequence(s) to your profile(s). It can align a variable number of sequences from 1 to N, with a variable number of profiles from 1 ot N: you can mix sequences and profiles in any proportion you like.

```
Adding one sequence to your MSA:
$$: t_coffee proteases_oneseq.fasta -profile proteases_small.aln

Adding many sequences to many profiles:
$$: t_coffee -profile=prf1.aln,prf2.aln,prf3.aln -outfile=combined_profiles.aln
```

Warning: You can also use all the methods you want but be aware when using external methods that profiles are nto always supported. When it is not, it is replaced with its consensus sequence which will not be quite as accurate. Methods supporting full profile information are: lalign_id_pair, slow_pair, proba_pair, clustalw_pair and clustalw_msa. All the other methods (internal or external) treat the profile as a consensus (less accurate).

5.4.7.2 Computing very accurate (but slow) alignments with PSI/TM-Coffee

PSI-Coffee is currently the most accurate mode of T-Coffee but also the slowest. Its principle is rather simple: it associates every sequence with a profile of homologous sequences gathered using BLAST on a sequence database (nr by default). PSI-Coffee then uses the profiles instead of the initial sequences to makes a multiple profile alignment (command 1). In a last step, your profiles are replaced by their initial query sequence from your initial dataset and returns a MSA of your sequences. PSI-Coffee can also use reduced database instead of nr (installed locally) in order

to speed-up the process. A special mode, TM-Coffee, exists using PSI-Coffee but specialized to align transmembrane proteins using a reduced database of TM proteins and also including a prediction of transmembrane domains with the flag **-template_file PSITM** (command 2). It is much faster as the search database is limited to known transmembrane protein, however, it applies in only specific cases unlike PSI-Coffee which is a general method. You can find more information about TM-Coffee here. If you want to specify a local BLAST version and a local database of your choice, just add to your command line the flags **-blast_server** and **-protein_db** and the corresponding paths.

```
Command 1: PSI-Coffee
$$: t_coffee sample_seq1.fasta -mode psicoffee

Command 2: TM-Coffee
##: t_coffee proteases_small.fasta -mode psicoffee -template_file PSITM -protein_db
↪<database>
```

Warning: PSI/TM-Coffee requires BLAST and a database to search; if you don't have BLAST installed locally, it will use the BLAST default of T-Coffee. More importantly, if you don't specify a reduced database for TM-Coffee, it will run on nr and be equal to PSI-Coffee.

Hint: When running PSI/TM-Coffee, T-Coffee will use the BLAST EBI by default; it can happen that the web service is unavailable from time to time, T-Coffee will return a warning asking you to use NCBI BLAST instead. You can either do that or wait and rerun your job later on.

5.5 Evaluating Your Alignment

There are three possible strategies for evaluating your alignment of protein sequences:

1) **Sequence based methods** like the CORE index and the TCS if you don't have any structure (quite often). These do pretty well in the core regions of an alignment (which can correspond to protein domains, fold, structural elements, etc...) but can be limited in more variable regions (which can correspond to loops, disordered regions, etc...).

2) **Structure is a killer**, so if you have at least two structures available for your protein family, you are in an ideal situation and you can use the iRMSD. If you only have one structure available, we developped STRIKE to compare alternative alignment.

3) **Another killer is the use of functional information**, but it is much less often at hand. If you know some residues MUST be aligned because they are functionally related. As the information is scarce and not standard, there is no automated procedure specifically designed for this kind of analysis but you can still setup an evaluation procedure with T-Coffee.

Note: Most of evalution methods are designed for protein sequences (notably structure based methods), however, T-Coffee via the TCS/CORE index offers some possibilities to evaluate also DNA alignments.

5.5.1 Sequence Based Methods

5.5.1.1 The CORE index

Note: The CORE index is the basis of T-Coffee estimation of the consistency, however for evaluating alignment we recommend to use the TCS procedure describe in the next section.

5.5.1.1.1 Computing the local CORE index

The CORE index is an estimation of the consistency between your alignment and the computed library. The higher the consistency, the better the alignment. The score reported with every T-Coffee alignment is the concistency score (depending on the version it can be normalized to 100 or 1000). If you want to go further and estimate the local concistency (known as the CORE index), an html file is automatically created each time you run T-Coffee; it is colored version of your alignment where residues are colored according to their consistency score, from blue (low consistency) to red (high consistency). It is not full proof but in principle you can expect positions with a score above 6 to be correctly aligned.

5.5.1.1.2 Computing the CORE index of any alignment

You can evaluate any existing alignment with the CORE index. All you need to do is provide that alignment with the -infile flag and specify that you want to evaluate it. For more information on filtering/trimming an alignment using the CORE index score, refer to the subsection **Preparing Your Data: Reformatting, Trimming an more. . . /Modifying the data itself**.

```
$$: t_coffee -infile=proteases_small_g10.aln -output=html -score
```

5.5.1.2 Transitive Consistency Score (TCS)

TCS is an alignment evaluation score that makes it possible to identify the most correct positions in an MSA. It has been shown that these positions are the most likely to be structuraly correct and also the most informative when estimating phylogenetic trees. The TCS evaluation and filtering procedure is implemented in the T-Coffee package and can be used to evaluate and filter any third party MSA (including T-Coffee MSA of course!).

Warning: TCS has been incorporated to T-Coffee recently, it means that not all distribution have the TCS implemented; you should install one the latest stable version of T-Coffee to have the TCS along with T-Coffee.

5.5.1.2.1 Evaluating an existing MSA

The TCS is most informative when used to identify low-scoring portions within an MSA. It is also worth noting that the TCS is not informative when aligning less than five sequences.

```
$$: t_coffee -infile sample_seq1.aln -evaluate -output=score_ascii,aln,score_html
```

Output files:

- `sample_seq1.score_ascii` displays the score of the MSA, the sequences and the residues.
- `sample_seq1.score_html` displays a colored version score of the MSA, the sequences and the residues.

Warning: The color code in the score_html indicates the agreement between the library and the considered alignment. It is important to understand that this score does not only depend on the input MSA, but it also depends on the library.

5.5.1.2.2 Filtering unreliable MSA positions

TCS allows you to filter out from your alignment regions that appears unreliable according to the consistency score; the filtering can be made at the residue level or the column level:

```
$$: t_coffee -infile sample_seq1.aln -evaluate -output tcs_residue_filter3, \
    tcs_column_filter3,tcs_residue_lower4
```

Output files:

- `sample_seq1.tcs_residue_filter3` All residues with a TCS score lower than 3 are filtered out
- `sample_seq1.tcs_column_filter3` All columns with a TCS score lower than 3 are filtered out
- `sample_seq1.tcs_residue_lower4` All residues with a TCS score lower than 3 are in lower case

Warning: The TCS will create output files containing your results; if you rerun similar jobs or with the same name, TCS will not overwrite the previous outputs but append the new results in the already existing files.

All these output functions are also compatible with the default T-Coffee (command 1) when computing an alignment or with **seq_reformat** (command 2) using a T-Coffee `<name>.score_ascii file`.

```
Command 1:
$$: t_coffee -seq sample_seq1.fasta -output tcs_residue_filter3, tcs_column_filter3, \
    tcs_residue_lower4

Command 2:
$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -struc_in sample_seq1.score_
↪ascii \
    -struc_in_f number_aln -output tcs_residue_filter3
```

5.5.1.2.3 Weighting MSA for improved trees

One cool trick about TCS is the possibility to deliver weigthed MSA, where each columns is multiplied according to its consistency score; this appears to be particularly useful to build phylogenetic tree. Phylogenetic trees are evaluated using a bootstrap score so each column as the same weight, no matter its relevance, in the case of weighted TCS, the more reliable columns are more represented, therefore improving the support of informative and reliable positions of your MSA.

```
$$: t_coffee -infile sample_seq1.aln -evaluate -output tcs_weighted, tcs_replicate_100
```

Output files:

- `sample_seq1.tcs_weighted` All columns are duplicated according to their TCS score
- `sample_seq1.tcs_replicate_100` Contains 100 replicates in phylip format with each column drawn with a probability corresponding to its TCS score

Note that all these output functions are also compatible with the default T-Coffee (command 1) when computing an alignment or with **seq_reformat** (command 2) using a T-Coffee .score_ascii file.

```
Command 1:
$$: t_coffee -seq sample_seq1.fasta -output tcs_weighted, tcs_replicate_100

Command 2:
$$: t_coffee -other_pg seq_reformat -in sample_seq1.aln -struc_in sample_seq1.score_
↪ascii \
    -struc_in_f number_aln -output tcs_weighted
```

5.5.1.2.4 Using different libraries for TCS

It is possible to change the way TCS reliability is estimated. This can be done by building different T-Coffee libraries. The proba_pair is the default aligner of T-Coffee that runs a pair-HMM to populate the library with residue pairs having the best posterior probabilities (command 1). You can also combine local and global alignners (command 2). There is also a fast alternative by using a special mode to run a series of fast multiple aligners; it is very fast and used by ENSEMBL Compara (command 3)

```
Command 1:
$$: t_coffee -infile sample_seq1.aln -evaluate -method proba_pair -output score_ascii,
↪ \
    aln, score_html

Command 2:
$$: t_coffee -infile sample_seq1.aln -evaluate -method proba_pair,lalign_id_pair \
    -output score_ascii,aln, score_html

Command 3:
$$: t_coffee -infile sample_seq1.aln -evaluate -method mafft_msa,kalign_msa,muscle_
↪msa \
    -output score_ascii, aln, score_html
```

5.5.1.2.5 Working with coding DNA

When working with DNA, it is advisable to first align the sequences at the protein level and later thread back the DNA onto your aligned proteins. The filtering must be done in two steps, as shown below. Note that your DNA and protein sequences must have the same name. This first step produces the TCS evaluation file `sample_prot_thread.score_ascii` (command 1). Then, the **-out dna.replicates** option produces 100 DNA replicates with positions selected according to their aminoacid TCS score (command 2). Finally, the **-out dna.filtered** option will filter the DNA alignment according to their TCS column score.

```
Command 1: evaluating the protein MSA
$$: t_coffee -infile sample_prot_thread.aln -evaluate -output score_ascii

Command 2: generating replicates for DNA
$$: t_coffee -other_pg seq_reformat -in sample_prot_thread.aln -in2 sample_dna_thread.
↪fa \
    -struc_in sample_prot_thread.score_ascii -struc_in_f number_aln -output tcs_
↪replicate_100 \
    -out dna.replicates

Command 3: filtering out according to the TCS score
```

(continues on next page)

(continued from previous page)

```
$$: t_coffee -other_pg seq_reformat -in sample_prot_thread.aln -in2 sample_dna_thread.
↪fa \
    -struc_in sample_prot_thread.score_ascii -struc_in_f number_aln -output tcs_
↪column_filter5 \
    -out dna.filter
```

5.5.1.2.6 Summary of the output options

Flags	Description
score_ascii	Outputs a TCS evaluation file
score_html	Contains ascii format in html format
score_pdf	Transfers score_html into pdf format
sp_ascii	Reports the TCS score of every aligned pair in the target MSA
tcs_residue_filter_N	Removes all residues with a TCS score lower than *N*
tcs_columns_filter_N	Removes all columns with a TCS score lower than *N*
tcs_weighted	Phylip format with duplicated columns according to their TCS score
tcs_replicate_N	Generates *N* replicates of columns drawn according to their TCS score

5.5.2 Structural evaluation of MSAs

5.5.2.1 APDB/iRMSD

5.5.2.1.1 What is the APDB/iRMSD?

APDB and the iRMSD are two closely related measures meant to evaluate the accuracy of a MSAt without using a structure based reference alignment. The iRMSD is a follow up of the APDB measure and we now recommend using the iRMSD rather than APDB. Although it may seem that the iRMSD was an attempt to get free iPODs from Apple, it is not (or at least we never got the iPODs). The iRMSD is a special RMSD (standing for intramolecular distances based RMSD) where the alignments are evaluated using the structural information of the sequences with known structures.

The strength of the iRMSD is its independence from a specific superposition models. When using the iRMSD to evaluate the score of a MSA, one does not need to superpose the two structures and deduce a sequence alignment that will then be compared with the target alignment. Given two aligned residues (X and Y) the iRMSD score is an attempt to estimate the neighborhood support for the XY alignment. This is done by measuring the difference of distances between X and Y and every other pair of aligned residues within the same sphere (W and Z). The iRMSD is obtained by measuring the average Root Mean Square (RMS) of these differences of distances. The first step of APDB/iRMSD is to measure the distances between the Ca (carbon alpha) of each residue and its neighbors. Neighborhood is defined as a sphere of radius ***-maximum_distance** (10 by default). However, by setting **-local_mode** to 'window', the sphere can be replaced with a window of 1/2 size **-maximum_distance** residues.

The lower the iRMSD, the better the alignment. Yet, an alignment can obtain a good iRMSD score by simply having few aligned residues. To avoid this, the program also reports a normalized version of the iRMSD, the NiRMSD= MIN(L1,L2)*iRMSD/Number considered columns. It is recommended to use the NiRMSD to compare alternative alignments of different length. From a structural point of view, the NiRMSD has a meaning very similar to the iRMSD and it behaves in a similar fashion from a numerical point of view (similar ranges in Angstroms).

Note: APDB is an older measure less robust than the iRMSD; it is an attempt to estimate the fraction of pairs of residues whose alignment seems to be correct form a structural point of view. The higher APDB, the better the alignment, and conversely the lower the NiRMSD, the better the alignment.

5.5.2.1.2 How to efficiently use structural information?

When it comes to evaluating MSAs, nothing is better than structural information. To use the methods we describe here, you will need to have at least two structures, similar enough (>60%) to sequences contained in your dataset. Here an outline of the best way to proceed:

1) Try to include two structures with distantly related sequences, the other sequences being intermediates.
2) Align your sequences without using the structural information (i.e. T-Coffee, MUSCLE, MAFFT...).
3) Evaluate your alignment with iRMSD/NiRMSD (see later in this section); give this alignment the score S1.
4) Realign your sequences but this time using structural information with Expresso.
5) Measure the score of that alignment; the score will be S2.

If S1 and S2 are almost similar, it means your distantly related structures were well aligned and you can expect the intermediate sequences to be well aligned as well. If S2 is much better than S1, you can expect the structures to be well aligned in the second alignment, while there is no guarantee that the alignment of the intermediate sequences has improved as well, although in practice it often does.

5.5.2.1.3 Evaluating an alignment with the iRMSD package

Let us evaluate the alignment produced by Expresso using the template file it returns. Evaluating the MSA with iRMSD will deliver a long output (all pairs are compared), the most interesting bit is at the bottom with the global iRMSD score in Angstrom (NiRMSD is the iRMSD score normalized to the length of the MSA).

```
Running the iRMSD:
$$: t_coffee -other_pg irmsd proteases_small.aln -template_file proteases_small.
↪template

Result of the iRMSD evaluation:
TOTAL     EVALUATED :  50.17 %
TOTAL     APDB      :  83.44 %
TOTAL     iRMSD     :  0.67 Angs
TOTAL     NiRMSD    :  1.34 Angs
```

5.5.2.1.4 Evaluating alternative alignments

The strength of structure based alignments is that they make it possible to compare alternative alignments. In this case let us consider the following results in the table below (APDB in %, iRMSD/NiRMSD in Angstroms, and the evaluated columns in %). As expected, Expresso delivers the best alignment from a structural point of view. This makes sense, since Expresso explicitely USES structural information. The other figures show us that the structural based alignment is only marginally better than most sequences based alignments, yet with the notable exception of ClustalW.

Method	APDB(%)	iRMSD(A)	NiRMSD(A)	Eval. (%)
Expresso	83.44	0.67	1.34	50.17
T-Coffee	83.11	0.68	1.35	50.29
M-Coffee	83.08	0.68	1.36	50.00
ProbCons	83.10	0.68	1.35	50.28
MAFFT	82.99	0.68	1.35	50.25
Kalign	82.42	0.69	1.38	50.02
ClustalW	80.62	0.73	1.47	49.55

5.5.2.1.5 DEC - Distance Evolutionnary Conservation with msa2distances

Using a similar approach it is possible to estimate the variation of intra-molecular distances across a multiple structural MSA for every pair of residue of every sequence. The following command also returns the average variation (stdev) for every residue and its neighbours withing <radius>

```
$$: t_coffee -other_pg seq_reformat  -in proteases_small.aln -in2 proteases_small.
↪template -action +msa2distances  20
#$: t_coffee -other_pg seq_reformat  -in proteases_small.aln -in2 proteases_small.
↪template -action +msa2distances  <radius: 20A> <threshold: 0.5 A> <min cluster: 4>
```

The output comes in two sections that can be grepped with the first keyword:

```
##DECPAIR s1:                ##DECPAIR s1:   <seq_name> c1:  <column 1> c2: <column 2>␣
↪r1:  <residue 1> r2:  <residue 2> pdbr1:  <pos filed on ATOM line> pdbr2:  pos␣
↪filed on ATOM line> aa1: <amino acid 1> aa2: <amino acid 2> d:  <distance> avg_d:
↪<average distance in the column> stdev_d:  <standard deviation> Normalized_score:
↪<standard_deviation/average_distance> N: <number of residue pairs> F: <number␣
↪residue pairs/number sequences> ent1: <shannon entropy col 1> ent2: <shannon␣
↪entropy col2>
```

The second fraction summarizes the fate of each residue. Note that the radius parameter potentially makes each residue in a column resulting having different levels of conservation

```
##DECRES s1:                <seq name> aa: <amino acid> c1: <msa column> r1:  <residue␣
↪index 1..N> avg_stdev: <average stdev across radius AA> -Zscore:  <average zscores␣
↪of the averaged stdev> normZ: <normalized zscore 0..100> Neighborhood:  <# of AA␣
↪within radius> Radius:  <radius size in Angstrom as specified in the command line,␣
↪20 A by default>
```

This function also outputs colorized versions of the input PDBs using the PDB Bfactor filed. The output files are the following:

5.5.2.2 STRIKE: Contact based evaluations

STRIKE uses a contact matrix to evaluate an alignment using one or more structures. When doing so, the contacts are extracted from the sequences with known structures and are tnen projected onto the other sequences. Each sequence is then evaluated for the quality of its predicted contacts. When more than one structure is available, the results are provided using each structure as a template. The following Command line will carry this analysis. It takes as input an MSA containing at least one sequence with one of the templates defiend in the template file. The PDB files must be within the current directory.

```
$$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.aln -in2 sample_3Dseq1.template␣
↪-action \
    +evaluate3D strike -output score_ascii -out sample_3D.score_ascii
```

The first part of the output is the STRIKE RAW SCORE. It reports the score of each sequence when using either one structure as a template, or averaging over all the structures; the highest value in every column is marked with a star. The various fields are as follows:

Field	Name	Definition
RS	Raw Score	Average STRIKE score
Rn	Random Score	Average score when scrambling contacts
Bg	Background Score	Average score of all-against-all contacts

It is also possible to use three extra parameters. The three parameters must be passed!!! Each one can be replaced with the string **"def"** if the default value is meant to be used.

Parameter	Default	Definition
max	1.2	Max distance between two contacting AA (Angstrom)
end	3	Number of excluded neighbours between contacts
matrix	strike	File containing the STRIKE matrix

```
Generic common line for STRIKE:
##: t_coffee -other_pg seq_reformat -in sample_3Dseq1.aln -in2 sample_3Dseq1.template␣
↪-action \
    +evaluate3D strike <max> <end> <matrix> -output score_ascii

Generic common line for colored output:
##: t_coffee -other_pg seq_reformat -in sample_3Dseq1.aln -in2 sample_3Dseq1.template␣
↪-action \
    +evaluate3D strike <max> <end> <matrix> -output score_html -out sample_3Dseq1.html
```

5.5.3 Evaluating a MSA according to your own criterion

Any kind of feature can easily be turned into an evaluation grid. For instance, the protease sequences we have been using here have a well characterized binding site. A possible evaluation can be made as follows: let us consider the UniProt annotation of the two distantly related sequences; these two sequences contain the electron relay system of the proteases. We can use it to build an evaluation library: in P29786 (TRY3_AEDAE) the Histidine residue is at position 68 while in P21844 (MCPT5_MOUSE) the functionally equivalent Histidine residue is at position 66. We can therefore build a library that will check whether these two residues are properly aligned in any MSA. The library will look like this:

```
! TC_LIB_FORMAT_01
2
sp|P21844|MCPT5_MOUSE 247 MHLLTLHLLLLLLGSSTKAGEIIGGTECIPHSRPYMAYLEIVTSENYLSACS\
GFLIRRNFVLTAAHCAGRSITVLLGAHNKTSKEDTWQKLEVEKQFLHPKYDENLVVHDIMLLKLKEKAKLTLGVGTLP\
LSANFNFIPPGRMCRAVGWGRTNVNEPASDTLQEVKMRLQEPQACKHFTSFRHNSQLCVGNPKKMQNVYKGDSGGPLL\
CAGIAQGIASYVHRNAKPPAVFTRISHYRPWINKILREN
sp|P29786|TRY3_AEDAE 254 MNQFLFVSFCALLDSAKVSAATLSSGRIVGGFQIDIAEVPHQVSLQRSGRHFC\
GGSIISPRWVLTRAHCTTNTDPAAYTIRAGSTDRTNGGIIVKVKSVIPHPQYNGDTYNYDFSLLELDESIGFSRSIEA\
IALPDASETVADGAMCTVSGWGDTKNVFEMNTLLRAVNVPSYNQAECAAALVNVVPVTEQMICAGYAAGGKDSCQGDS\
GGPLVSGDKLVGVVSWGKGCALPNLPGVYARVSTVRQWIREVSEV
#1 2
66 68 100
! SEQ_1_TO_N
```

You simply need to cut and paste this library in a file and use this file to measure the concistency between your alignment and the correspondances declared in your library. The following command line also makes it possible to visually display the agreement between your sequences and the library.

```
$$: t_coffee -infile proteases_small.aln -lib charge_relay_lib.tc_lib -score \
    -output html
```

5.6 Downstream Analysis

5.6.1 Contact Evaluations

5.6.1.1 Contact Evaluation on proteins

This function will report in a library format all the intra-molecular distances of the provided sequences

```
Command 1:
$$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.aln -in2 sample_3Dseq1.template␣
↪-action +pdb2contacts intra  distances 10 -output contact_lib
```

This command will report all the distances between every pair of residue less than 10 Angstrom appart. In the output every pair of residue will be assocuiated with an integer value equal to 100*distance in Angstrom. Distances are between CA

Command 1: $$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.aln -in2 sample_3Dseq1.template -action +pdb2contacts intra contacts 1.2 -output contact_lib

This command will report all the pairs of residue featuring two atoms less than 1.2 Angstrom appart

5.6.2 Clustering/Trees based on protein 3D structures

This section describes tree estimation procedure based on the comparison of intramolecular distances. These methods are slightly different from the *T-RMSD <http://tcoffee.readthedocs.io/en/latest/tcoffee_main_documentation.html#the-t-rmsd>* that was also originally developped for this purpose and whose usage is decribed in the next section. The methods described in this section are more versatile than the original T-RMSD: they support trees based on contacts or intra-molecular distance variations,alternative tree reconstruction algorithms (NJ, UPGMA), genuine column based generalized boostrap procedures. They also support the STRIKE protocol that involves projecting contacts measured within one or more experimental structures onto the sequences these are aligned with and using the *STRIKE <https://academic.oup.com/bioinformatics/article/27/24/3385/306640/STRIKE-evaluation-of-protein-MSAs-using-a-single>* contact matrice to score these contacts in the sequences without known structures.

5.6.2.1 Generating a tree based on 3D intramolecular distances conservation

This option makes it possible to estimate a tree while taking into account the variation of intramolecular distances within the considered sequences. It requires in input an alignment (FASTA, MSF, ClustalW...) and a template file (structure files associated with the query sequences). The following command (command 1) will generate a 100 replicate NJ trees using the difference of distances between pairs of aligned residues, at a maximum cut-off of 15A. Columns with less than 50% residues are ignored. It is possible to control the defautl parameters (command 2). The ouput will be a tree in the Newick format with bootstrap supports.

```
Command 1:
$$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.aln -in2 sample_3Dseq1.template␣
↪-action \
    +tree replicates 100 +evaluate3D distances +tree2bs first +print_replicates -
↪output newick
```

This command will print the main tree with bootstrap support values (first line). The 100 replicates (+tree replicates 100) will then be printed in the following lines (+print_replicates flag)

```
Command 2:
$$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.aln -in2 sample_3Dseq1.template␣
↪-action \
    +tree replicates 100 +evaluate3D distances +tree2bs first +print_replicates -
↪output dm
```

This command will print the main distance matrix in triangular format (first block). The 100 replicates (+tree replicates 100) will then be printed in the following blocks (+print_replicates flag)

```
Command 3:
$$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.aln -in2 sample_3Dseq1.template␣
↪-action \
    +tree replicates 100 +evaluate3D distances +tree2bs first +print_replicates -
↪output newick_dm
```

This command will print the newick trees and their corresponding distance matrix in triangular format (first block). The 100 replicates (+tree replicates 100) will then be printed in the following blocks along with the trees (+print_replicates flag). Note that the lines containing trees can be grepped by the ";" symbol they terminate with.

```
Command 4:
$$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.aln -in2 sample_3Dseq1.template␣
↪-action \
    +tree replicates 100 gap 0.5 mode nj +evaluate3D distances 15 +tree2bs first -
↪output newick
```

This command will do the same but pairs of residues will only contribute to the distance computation if they are less than 15 Angstroms appart (i.e. when comparing two aligned pairs, both pairs of residues in each sequence must be less than 15 Angstroms appart)

Warning: Sequences without 3D structure will be excluded from the analysis and from the final output.

5.6.2.2 Generating a tree based on contact conservation

The following option (command 1) makes it possible to estimate a tree while taking into account the variation of contact conservation within the considered sequences. This call will generate a 100 replicate NJ trees using as a distance metrics the fraction of contacts conserved between pairs of aligned residues, at a maximum cutoff of 1.2 A between Van der Waals radius and ignoring the 3 closest neighbors; columns with less than 50% residues are ignored. For sequences without 3D information, the STRIKE contact potential is used instead (Watson and Crick base pairing propensity for RNA). The output will be a tree in Newick format. It is possible to control default parameters using the following extended command line (command 2).

```
Command 1:
$$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.aln -in2 sample_3Dseq1.template␣
↪-action \
    +tree replicates 100 +evaluate3D contacts +tree2bs first -output newick -out tree.
↪dnd

Command 2:
$$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.aln -in2 sample_3Dseq1.template␣
↪-action \
    +tree replicates 100 gap 0.5 mode nj +evaluate3D contacts 1.2 3 +tree2bs first -
↪output \
    newick -out tree.dnd
```

Warning: The procedure requires at least 1 sequence with a known 3D structure or with contact information.

5.6.2.3 Visualizing contact/distance conservation

This same procedure can be used to visualize either intramolecular distance conservation or contact conservation. The output option **score_raw** generate a tabulated dump of the numerical values associated with every residue, sequence and column of the considered alignment. The flag **-out** specifies the name of the output file containing the results (you can give the name you want, but if you read the documentation so far, you already know it...).

```
$$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.aln -in2 sample_3Dseq1.template␣
↪\
    -action +evaluate3D distances -output score_html -out out.html

$$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.aln -in2 sample_3Dseq1.template␣
↪\
    -action +evaluate3D distances -output score_ascii -out out.ascii

$$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.aln -in2 sample_3Dseq1.template␣
↪\
    -action +evaluate3D distances -output score_raw -out out.tab
```

5.6.2.4 Visualizing informative positions

If you have a well defined subgroup of sequences (domains having the same function, same specificity, etc...), it is possible to estimate which columns yield the best support using the following command. The input are an alignment of your sequences, a template file containing the list of structure files to use as templates and a FASTA file of the sequences that form the group whose support you want to analyze. The output will be a colored version of your MSA indicating the sequences that best contribute to your clustering.

```
$$: t_coffee -other_pg seq_reformat -in sample_3Dseq1.aln -in2 sample_3Dseq1.template␣
↪\
    -action +tree replicates columns +evaluate3D  distances +evaluateTree group_
↪3Dseq1.fasta \
    -output score_html -out out_aln.html
```

5.6.2.5 Evaluating clustering capacities (under maintenance...)

If you want to check the capacity of an algorithm to bring related sequences within monophyletic groups, you should name your sequences according to the group they belong to (XXXX_1 for members of group 1; YYYY_2, for members of group 2; etc...) and use the following evaluation procedure. The output will be the number of monophyletic groups containing sequences belonging to the same group. The tree can be precomputed (command 1) or it can be computed on the fly (command 2).

```
Command 1: precomputed tree
##: t_coffee -other_pg seq_reformat -in <tree> +tree2collapse groups 4 +print nseq -
↪output no

Command 2: computed on the fly
##: t_coffee -other_pg seq_reformat -in <aln> -in2 <template> -action +tree␣
↪replicates 100 \
    +evaluate3D  distances 15 +tree2bs first +tree2collapse groups 4 +print nseq -
↪output no
```

5.6.2.6 Structural Tree Parameters

The structural tree parameters come through three distinct flags:

- **-tree** :Used to pass Tree computation parameters
- *** replicates <I=integer> *** : Indicates how many replicates I are to be produced (def=1) when doing the bootsrap
- *** mode <nj|upgma> *** : Tree computation mode to be used
- *** gap <F=float>*** : ignores columns with a fraction of gaps higher or equal to F (def=0.5)
- **-evaluate3D contacts <MaxD> <Ng>**:Used to estimate the cointact matrix
- ***<MAxD=float>*** : maximum distance between the closest atoms (including side chains) of two residues <def=1.3>
- ***<Ng=integer>*** : Number of excluded neighbours (on both side) <def=3>
- ***def*** : Any value can be left to its default as follows -evaluate3D contacts def 5 will used 1.2 as maxD and 5 as neighbor cutoff
- **-evaluate3D distances <MaxD> <Ng>**:Used to estimate the cointact matrix
- ***<MAxD=float>*** : maximum distance between the closest atoms (including side chains) of two residues <def=15>. When set to 0, all against all distances are computed.
- ***<Ng=integer>*** : Number of excluded neighbours (on both side) <def=3>
- ***def*** : Any value can be left to its default as follows -evaluate3D contacts def 5 will used 1.2 as maxD and 5 as neighbor cutoff
- **-evaluate3D strike <MaxD> <Ng> <mat>**:Used to estimate the cointact matrix
- ***<MAxD=float>*** : maximum distance between the closest atoms (including side chains) of two residues <def=15>
- ***<Ng=integer>*** : Number of excluded neighbours (on both side) <def=3>
- ***<mat=string>*** : strike matrix file (def="strike")
- ***def*** : Any value can be left to its default as follows -evaluate3D contacts def 5 will used 1.2 as maxD and 5 as neighbor cutoff

Distance based trees measures are based on a normalized difference of distances between pairs of residues:

```
X AB
Y CD
d1=d(A,B)
d2=d(B,B)
score(AB, CD)=(Min (d1/d2, d2/d1)^3)*d1
score (X,Y)=Sum(score(AB,CD))+Sum(score (CD, AB)/Sum (d(AB))+Sum (d(CD)) over all␣
↪pairs for which d(AB)>=MaxD AND d(CD)>=MaxD
```

5.6.3 The T-RMSD

5.6.3.1 What is the T-RMSD?

T-RMSD (Tree based on Root Mean Square Deviation) is a structure based clustering method using the iRMSD to drive the structural clustering of your aligned sequences with available structures. The T-RMSD supports all the parameters supported by iRMSD or APDB. T-RMSD is a distance RMSD (dRMSD) based method which generate

Structural Trees (analogue to phylogenetic tree) to determine fine-grained structural variations associated with a given Multiple Sequence Alignment (generated by the user using the method of is choice). The specificity of T-RMSD compared to other structural comparison methods stems from its capacity to generate a structural tree with values equivalent bootstrap values supporting the structural clustering. Such clustering is achieved by the construction of matrixes of distances, calculated between equivalent residues as defined by the ungapped columns of the given MSA. The resulting matrixes are then combined using the CONSENSE program from the Phylip package to generate a consensus structural tree with equivalent bootstrap values supporting each node (from 0 to 100; 100 indicating that all positions support the clustering).

5.6.3.2 Generating a structural tree with support values

T-RMSD is a special modof T-Coffee. To run T-RMSD, you just need a MSA (generated the way you want) and a template file; you need one structure for each sequence, otherwise these sequences will be excluded from the final results. There are several output files:

- `<input name>.struc_tree.list`: list of all individual trees (one per ungapped column)
- `<input name>.struc_tree.consense_output`: basic consensus tree rendering
- `<input name>.struc_tree.consensus`: resulting structural tree with support (Newick)
- `<input name>.struc_tree.html`: colored MSA according to the contribution to the clusterin#

```
$$: t_coffee -other_pg trmsd -aln sample_3Dseq1.aln -template_file sample_3Dseq1.
↪template
```

Hint: Another important information displayed only on screen is the number of usable positions used to build the clustering. If your alignment is too gappy, this percentage will be low and the resulting clustering may not be accurate nor informative; in the previous example, 28.57% of the MSA will be used.

5.6.3.3 Visualizing the structural clustering

T-Coffee have limited tree visualization capacities, yet the T-RMSD delivers two relevant files in this matter: the first one is a basic rendering of the tree with the support values for each node (`<input name>.struc_tree.list`), the second one is a colored html of the MSA where columns are colored according to their individual support to the final consensus tree (`<input name>.struc_tree.html`). If you want to visualize/modify the resulting tree, we recommend to use a dedicated tree viewer such as PhyloWidget. By default PhyloWidget is used by the T-RMSD web server, but as the format is standard (Newick format) you can use any viewer you want (iTOL, ETE, FigTree, Phylodendron, etc...)

5.6.4 The TCS

The TCS (Transitive Consistency Score) is a measure of consistency between a multiple sequence alignmment and a library of pairwise alignments of the same sequences. In the original *publication <https://academic.oup.com/mbe/article/31/6/1625/2925802/TCS-A-New-Multiple-Sequence-Alignment-Reliability>* the TCS score was shown to correlate well with local structural accuracy and with the phylogenic reconstruction potential of individual MSA columns. As such the TCS can therefore be used to down-weight or filter out the less reliable positions. The TCS is available as *>web-server http://tcoffee.crg.cat/apps/tcoffee/do:core>* and its command line usage is described in details in an *earlier section <http://tcoffee.readthedocs.io/en/latest/tcoffee_main_documentation.html#transitive-consistency-score-tcs>* of this documentation.

It can be used to evaluate and compare existing alignmnets . The original publication describing it is available described in the previous subsection as it is mainly an evaluation tool. It has however several option that are relevant for downstream analysis such weighting and/or trimming MSA for phylogenetic reconstruction. It was demonstrated that TCS was able to improve phylogenetic reconstruction, you are welcome to look at the publication for more details.

5.7 Internal/External Methods

Note: The real power of T-Coffee is its ability to seamlessly combine many methods into one. While we try to integrate as many methods as we can in the default distribution, we do not have the means to be exhaustive and if you desperately need your favorite method to be integrated, you will need to bite the bullet ...

5.7.1 What are the methods already integrated in T-Coffee?

Although, it does not necessarily do so explicitly, T-Coffee always end up combining libraries (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 being 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 to generate constraints. 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 available method in T-Coffee, and how you can add your own methods either through direct parameterization or via a perl script. There are two kinds of methods: the internal and the external. For the internal methods, you simply need to have T-Coffee up and running. The external methods are generally installed with T-Coffee (when using the installer), but if you have problems with some packages, refer to the **T-Coffee Installation** chapter.

5.7.1.1 INTERNAL methods (built-in)

Internal methods can be requested using the following names:

- **proba_pair**: adapted from ProbCons, this method (the current default) uses a pair HMM to compute a pairwise alignment with a biphasic gap penalty.
- **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**: iterative fast_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 result is therefore dependent 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. [UNSUPPORTED]

- **lalign_id_pair**: uses the ten top non intersecting local alignments, as delivered by lalign. Each alignement is weighted with its average percent identity.
- **lalign_rs_s_pair**: Same as above but does also does self comparison and uses the lalign raw_score (s stands for self). This is needed when extracting repeats. [UNSUPPORTED]
- **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.
- **cdna_fast_pair**: this method computes the pairwise alignment of two cDNA sequences. It is a fast_pair alignment that only takes into account the aminoacid similarity with different penalties for insertions and frameshifts. This alignment is turned into a library where matched nucleotides receive a score equal to the average level of identity at the aminoacid level. This mode is intended to clean cDNA obtained from ESTs or to align pseudogenes. [UNSUPPORTED]

5.7.1.2 EXTERNAL methods (plug-in)

External methods correspond to packages developed 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 add an extra interface. To have a complete list of the methods that can be used as plug-ins, just type **t_coffee** in a terminal, it will we displayed on your screen. Most of these methods can be used as either pairwise (**<method>_pair**) or multiple alignment methods (**<method>_msa**); note that all these methods use Blosum62 as a default.

One package is a bit different; **fugue_pair** uses a standard FUGUE installation to make a sequence/structure alignment. Installation must be standard but it does not have to include all the FUGUE packages but only:

```
Required packages:
1) joy, melody, fugueali, sstruc, hbond
2) copy fugue/classdef.dat /data/fugue/SUBST/classdef.dat

Configuration:
##: setenv MELODY_CLASSDEF=<location>
##: setenv MELODY_SUBST=fugue/allmat.dat
```

5.7.2 Modifying the parameters of INTERNAL/EXTERNAL methods

5.7.2.1 Internal methods

It is possible to modify on the fly the parameters of hard coded methods (**EP** stands for Extra parameters). These parameters will superseed any other parameters.

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

5.7.2.2 External methods

External methods receive a command line built with the information provided via the parameter file (see next heading). It is possible to produce such a parameter file and to modify it in order to modify the commands passed to the methods. The passed command is built as follows:

```
##: <EXECUTABLE><PARAM1><IN_FLAG><seq_file><PARAM2><OUT_FLAG><outname><PARAM>
```

You should know what is the best place for squizing your extra parameters. It will depend on the application, although PARAM2 is usually a good guess. Now if you want, for instance to modify the gap penalty of clustalw, you can try the following (of course, you must know the command line of the program you are trying to modify (ClustalW in this case):

```
$$: t_coffee sample_seq1.fasta -method clustalw_msa@EP@PARAM2@-GAPOPEN%e100%s-GAPEXT
↪%e10

<@EP>     : indicates that you will pass an extra parameter
<@PARAM1> : is the name of this parameter
<%s>      : replaces spaces
<%e>      : replaces the equal sign
```

5.7.3 Integrating EXTERNAL methods

If the method you need is not already included in T-Coffee, you will need to integrate it yourself. We give you here some guidelines on how to do so.

5.7.3.1 Accessing external methods

A special method exists in T-Coffee that can be used to invoke any existing program (command 1): for instance, ClustalwWis a method that can be ran with the command 2. Here is just an example but 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** for external method) uses a particular syntax (command 3).

```
Command 1: T-Coffee using ClustalW
$*: t_coffee sample_seq1.fasta -method=em@clustalw@pairwise

Command 2: ClustalW command
##: method -infile=<infile> -outfile=<outfile>

Command 3: Syntax for EXTERNAL methods
##: em@<method>@<aln_mode:pairwises_pairwise|multiple>
```

5.7.3.2 Customizing an external method

T-Coffee can run external method using a `tc_method` file that can be used in place of an established method. Two such files are incorporated in T-Coffee (`clustalw_method.tc_method` and `generic_method.tc_method`). You can dump them and customize them according to your needs. The first file is a very straightforward example on how to run Clustalw via T-Coffee with a set of parameters you may be interested in. Note that **ALN_MODE** instructs T_Coffee to run ClustalW on every pair of sequences. The second file is detailed in the next section.

```
1) Getting the configuration file:
$$: t_coffee -other_pg unpack_clustalw_method.tc_method

2) Format of the configuration file:
*TC_METHOD_FORMAT_01

***************clustalw_method.tc_method*********
EXECUTABLE clustalw
```

(continues on next page)

(continued from previous page)

```
ALN_MODE pairwise
IN_FLAG -INFILE=
OUT_FLAG -OUTFILE=
OUT_MODE aln
PARAM -gapopen=-10
SEQ_TYPE S
**********************************************

3) The configuration file will cause T-Coffee to emit the following system call:
##: clustalw -INFILE=tmpfile1 -OUTFILE=tmpfile2 -gapopen=-10

4) Running ClustalW via T-Coffee (in your working DIR):
$$: t_coffee sample_seq1.fasta -method clustalw_method.tc_method
```

5.7.3.3 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 which contains many hints on how to customize your new method. The `tc_method` files are treated like any standard established method in T-Coffee.

```
1) Using any generic method
$$: t_coffee -other_pg unpack_generic_method.tc_method

2) Format of the configuration 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: &amp;bsnp can be used to for white spaces

3) Run the method:
$*: t_coffee sample_seq1.fasta -method generic_method.tc_method
```

T-Coffee runs the script **tc_generic_method.pl** on your data. It also provides the script with parameters. In the case **-method clustalw** indicates that the script should run ClustalW on your data. Over the time, this script will be the place where novel methods will be integrated and make it possible to run any available method. It will be used to run the script **tc_generic_method.pl**, a perl script automatically generated by T-Coffee.

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.

5.7.3.4 The mother of all method files...

Here is the mother of all configuration file for T-Coffee:

```
*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>: &amp;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 (Library)
* fA -> Internal Function returning an Alignmnent
*
OUT_MODE aln
*
```

(continues on next page)

(continued from previous page)

```
*********************************************************
* IN_FLAG *
*********************************************************
*IN_FLAG
*flag indicating the name of the in coming sequences
*IN_FLAG no_name ->no flag
*IN_FLAG &amp;nbsp-in&amp;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 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 *
*********************************************************
```

5.7.3.5 Weighting your method

By default, the alignment produced by your method will be weighted according to its percent identity. However, this can be customized via the **WEIGHT** parameter, which supports all the values of the **-weight** flag. The only difference is that the **-weight** value thus declared will only be applied onto your method. If needed you can also modify on the fly the WEIGHT value of your method:

```
Increases by a factor 2 the weight of slow_pair:
$$: t_coffee sample_seq1.fasta -method slow_pair@WEIGHT@OW2
```

(continues on next page)

(continued from previous page)

```
Causes every pair of slow_pair to have a weight equal to 250:
$$: t_coffee sample_seq1.fasta -method slow_pair@WEIGHT@250
```

5.7.3.6 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 environement variable **METHODS_4_TCOFFEE****.

5.7.4 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.

5.7.4.1 Using precomputed 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. You can even combine multiple MSAs (refers to subsection **Combining multiple MSAs**).

5.7.4.2 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 (command 1) or convert your alignment into a library (command 2), or use the **-weight** flag (command 3).

```
Command 1:
$$: t_coffee -aln=sample_aln1.aln,sample_aln2.aln -method=fast_pair,lalign_id_pair

Command 2:
$$: t_coffee -aln sample_aln1.aln -lib=test_lib.tc_lib

Command 3:
$$: t_coffee -aln sample_aln1.aln -out_lib=test_lib.tc_lib -lib_only -weight=sim_
↪pam250mt
```

5.7.4.3 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 precompute your libraries in order to save them for further use. For instance, in the following example, we generate the local (command 1) and the global (command 2) libraries and later reuse them for combination into a multiple alignment. Once these libraries have been computed, you can then combine them at your convenience in a single MSA (command 3). Of course you can decide to only use the local or the global library.

```
Command 1:
$$: t_coffee sample_seq1.fasta -method lalign_id_pair -out_lib lalign_id_pair_seq1.tc_
↪lib \
    -lib_only

Command 2:
$$: t_coffee sample_seq1.fasta -method slow_pair -out_lib slow_pair_seq1.tc_lib -lib_
↪only

Command 3:
$$: t_coffee sample_seq1.fasta -lib lalign_id_pair_seq1.tc_lib, slow_pair_seq1.tc_lib
```

5.7.5 Plug-out: using T-Coffee as a plug-in

Just because it enjoys enslaving other methods as plug-in does not mean that T-Coffee does not enjoy being incorporated within other packages. We try to give as much support as possible to anyone who wishes to incorporate T-Coffee in an alignment pipeline. If you want to do so, please work out some way to incorporate T-Coffee in your script. If you need some help along the way, do not hesitate to ask, as we will always be happy to either give assistance or even modify the package so that it accomodates as many needs as possible. Once that procedure is over, set aside a couple of input files with the correct parameterisation and send them to us. These will be included as a distribution test, to insure that any further distribution remains compliant with your application.

CHAPTER 6

T-Coffee Technical Documentation

Warning: This chapter has been extensively updated in 11/2016. The **T-Coffee Technical Documentation** covers the different parameters, flags and options. Everything described here should be working from T-Coffee version 9.03 and higher, yet there is a trade-off between new options present in the most recent versions and old options now deprecated, unsupported or substituted by new ones (don't worry it will specified in the documentation).

Warning: T-Coffee is not POSIX compliant (sorry !!!).

6.1 T-Coffee Parameters & Flags

6.1.1 T-Coffee general information

6.1.1.1 General syntax

About the syntax of T-Coffee command lines, just for you to know that T-Coffee is quite flexible, 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 completion provided by many shells, you can replace '=' and ',' with a space.

6.1.1.2 T-Coffee flags

This documentation 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. When running T-Coffee, some options and their associated values or parameters will be displayed on screen (command 1). You can also display the list of all the flags (command 2) or a single flag (command 3) used in the version of T-Coffee you are using along with their default value type.

```
Command 1:
$$: t_coffee

Command 2:
$$: t_coffee -help

Command 3:
$#: t_coffee -help -<flag>
```

6.1.1.3 Setting up the parameters

There are many ways to enter parameters in T-Coffee (see the **-parameters** flag). In general you won't 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 reusing a specific set of parameters, different from those in `t_coffee_default`, hence the possibility to write an extra parameter file with the flag **-parameters**. In summary, this means that **-parameters** supersede all the other options, while parameters provided via **-mode** are the weakest.

Hint: Priorities: "-parameters" > "prompt parameters" > "-t_coffee_defaults" > "-mode"

6.1.1.4 Setting up the variables

It is possible to modify T-Coffee's behavior by setting any of the following environment variables. With the bash shell, use **export VAR='value'**; with the cshell, use **set $VAR='value'**. Here is a list of variables in T-Coffee:

- **http_proxy_4_TCOFFEE**: sets the http_proxy and HTTP_proxy values used by T-Coffee. These values supersede http_proxy and HTTP_proxy while http_proxy_4_TCOFFEE gets superseded by the command line values **-proxy** and **-email**; if you have no proxy, just set this value to an empty string.
- **email_4_TCOFFEE**: sets the E-mail values provided to web services called upon by T-Coffee; can be overriden by the flag **-email**.
- **DIR_4_TCOFFEE**: by default this variable is set to $HOME/.t_coffee; this is where T-Coffee expects to find its cache, tmp dir and possibly any temporary data stored by the program.
- **TMP_4_TCOFFEE**: by default this variable is set to $HOME/.t_coffee/tmp; this is where T-Coffee stores temporary files.
- **CACHE_4_TCOFFEE**: by default this variable is set to $HOME/.t_coffee/cache; this is where T-Coffee stores any data expensive to obtain: PDB files, structural alignments...
- **PLUGINS_4_TCOFFEE**: by default all the third party packages are searched in the directory DIR_4_TCOFFEE/plugins/<OS>. This variable overrides the default but can also be overriden by the **-plugins** flag.
- **NO_ERROR_REPORT_4_TCOFFEE**: by default this variable is no set; set it if you do not want the program to generate a verbose error output file (useful for running a server).
- **PDB_DIR**:indicate the location of your local PDB installation.
- **NO_WARNING_4_TCOFFEE**: suppresses all the warnings.

- **UNIQUE_DIR_4_TCOFFEE**: sets all DIR_4_TCOFFEE, CACHE_4_TCOFFEE, TMP_4_TCOFFEE, PLUGINS_4_TCOFFEE

T-Coffee can have its own environment file, kept in a file named `t_coffee_env` in the folder $HOME/.t_coffee/ and can be edited (under maintenance...). The value of any legal variable can be modified through that file. For instance, here are some examples of 1) **using a configuration file** when not requiring a proxy, 2) **setting up any environment variable** using the **-setenv** or 3) simply **using an export**.

```
1) No proxy
##: http_proxy_4_TCOFFEE=
##: EMAIL_4_TCOFFEE=cedric.notredame@gmail.com

2) Using 'setenv'
##: t_coffee ... -setenv ENV_4_TCOFFEE=<location>

3) Using 'export'
##: export ENV_4_TCOFFEE=<location>
```

Hint: Priorities: "export" > "-setenv" > "-proxy,-email" > "t_coffee_env" > "default environment"

Note: When you use **-setenv** for PATH, the value you provide is concatenated at the beginning of the current PATH value. This way you can force T-Coffee to use a specific version of an aligner.

6.1.1.5 CPU control

6.1.1.5.1 Multithreading

- **-multi_core** (usage:**-multi_core=[templates,jobs,relax,msa]**/default:none)

Specifies that T-Coffee should be multithreaded or not; by default all relevant steps are parallelized. The different options for the flag are the following:

- template: fetch the templates in a parallel way
- jobs: compute the library
- relax: extend the library in a parallel way
- msa: compute the msa in a parallel way
- no: not parallelized

- **-n_core** (usage:**-n_core= [number of cores+]**/default:1, deprecated: use -thread instead)

Use 0 to use all the cores - **-max_n_core** (usage:**-max_n_core= [number of cores+]**/default:1, deprecated: use -thread instead) Use 0 to use all the cores - **-thread** (usage:**-thread= [number of cores+]**/default:1) Use 0 to use all the cores

6.1.1.5.2 Limits

- **-maxlen** (usage:**-maxlen=[value,0=nolimit]**/default:**-maxlen=1000**)

Indicates the maximum length of the sequences.

- **-maxnseq** (usage:**-maxnseq=[value,0=nolimit]**/default:**-maxnseq=??**)

Indicates the maximum number of the sequences.

- **-ulimit** (usage:**-ulimit=[value]**/default:**-ulimit=0**)

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

- **-mem_mode** [Deprecated]

6.1.1.6 Meta-parameters

6.1.1.6.1 Global parameters

- **no flag**

If no flag is provided, your sequence dataset must be the first argument. When you do so, the name of your file is used as a name prefix for every output file of the program (changing the extension according to the type of result).

- **-mode**

A T-Coffee mode is a hard coded command line calling to specific options predetermined and optimized. By default, they are not used and should be called upon. Here are some examples: **expresso, mcoffee, rcoffee, evaluate, accurate, procoffee**... These modes have been designed to deliver the best results possible for a specific task; they can work without any parameters but can be controlled and modified extensively with extra parameters.

- **-parameters**

The input has to be a file containing extra parameters for T-Coffee. Parameters read this way behave as if they had been added on the right end of the command line that they either supersede one parameter or complete list of parameters. Here is an example (command 1) that will cause T-Coffee to apply the **fast_pair** method onto the sequences contained in `sample_seq1.fasta`. If you wish, you can also pipe these arguments into T-Coffee (command 2) by naming the parameter file 'stdin' (as a rule, any file named stdin is expected to receive its content via the stdin).

Warning: The parameter file can ONLY contain valid parameters; comments are not allowed. Parameters passed this way will be checked like normal parameters.

```
Command 1: defining parameters for T-coffee
$$: t_coffee -parameters=sample_param_file.param

Command 2: sending parameters to T-Coffee
$$: cat sample_param_file.param | t_coffee -parameters=stdin

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

- **-t_coffee_defaults**

The input has to be a file; it will 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**

- **-evaluate**

Replaces the former flag **-score** which is no longer supported. This flag toggles on the evaluate mode and causes T-Coffee to evaluate a precomputed MSA provided via **-infile=<MSA>**. The main purpose of this flag is to let you control every aspect of the evaluation, yet it is advisable to use predefined parameterization **-mode=evaluate**. The flag **-output** must be set to an appropriate format (refer to the subsection 'Alignments Flags').

```
$$: t_coffee -infile=sample_aln1.aln -mode=evaluate -method proba_pair

$$: t_coffee -infile=sample_seq1.aln -in Lsample_seq1_lib1.tc_lib -mode=evaluate
```

- **-convert** [cw]

By default, is turned off. It 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 option is ClustalW compliant.

6.1.1.6.2 Misc parameters

- **-version**

Returns the current version number of T-Coffee you are using.

- **-proxy**

Sets the proxy used by **HTTP_proxy** and **http_proxy**. Setting with the propmpt supersedes ANY other setting. Note that if you use no proxy, you should still set **-proxy**.

- **-email**

Sets your email value as provided for web services.

- **-cache** (usage:**-cache=[use, update, ignore, <filename>]**/default:none)

By default, T-Coffee stores in a cache directory the results of computationally expensive (structural alignment for instance) or network intensive operations (BLAST search). The usage is the following:.

- **-update**

Causes a wget access that checks whether the T-Coffee version you are using needs updating.

- **-plugins**

The input parameter has to be the directory, where all third pirty packages used by T-Coffee are kept (~/.t_coffee/plugins/ by default). As an alternative, you can also set the environment variable **PLUGINS_4_TCOFFEE** to your convenience.

- **-other_pg** (usage:**[seq_reformat,aln_compare,extract_from_pdb,irmsd,trmsd...]**)

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.

```
$$: t_coffee -other_pg=seq_reformat
$$: t_coffee -other_pg=unpack_all
```

- **-check_configuration** [under evaluation]

Checks your system to determine if all the programs T-Coffee can interact with are installed or not.

- **-full_log** [under evaluation]

Requires a file name as parameter; it causes T-Coffee to output a full log file that contains all the input/output files.

6.1.1.6.3 Verbose parameters

- **-quiet** (usage:**-quiet=[stderr,stdout,file name OR nothing]**/default:**-quiet=stderr**)

This control the verbose mode of T-Coffee from the display on the screen or to redirect to a given file; **-quiet** on its own redirect the output to /dev/null.

- **-no_warning** (usage:**-no_warning=[yes,no]**/default:none)

Suppresses all warning output of the verbose mode.

6.1.2 Input(s)

6.1.2.1 The "-in" flag

The **-in** flag and its identifier TAGs **are the real grinder of T-Coffee**. Sequences, methods, alignments, whatever... all pass through so that T-Coffee can turn them 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. Precompiled constraint list can also be provided. Each file provided via this flag must be preceded with a symbol (the identifier TAG) that indicates its nature to T-Coffee. The common usage is **-in=[<P,S,A,L,M,X><name>]**. By default it is set up to **-in=Mlalign_id_pair,Mclustalw_pair**. This is a legal multiple alignments that will be treated as single sequences (the sequences it contains will not be realigned). The TAGs currently supported are the following:

```
P : PDB structure
S : Sequences (aligned or unaligned sequences)
M : Methods used to build the library
L : Precomputed T-Coffee library
A : Alignments that must be turned into a Library
X : Substitution matrices
R : Profiles
```

If you do not want to use the TAGS, you will need to use the following flags in replacement. Do not use the TAGS when using these flags.

```
-aln     : Alignments  (A)
-profile : Profiles    (R)
-method  : Method      (M)
-seq     : Sequences   (S)
-lib     : Libraries   (L)
```

Note: The flag **-in** can be replaced with the combined usage of -aln, -profile, -pdb, -lib, -method depending on what you want.

```
$$: t_coffee -in=Ssample_seq1.fasta,Asample_seq1_aln2.aln,Asample_seq1_aln2.msf, \
    Mlalign_id_pair,Lsample_seq1_lib1.tc_lib -outfile=outaln
```

This command will trigger the following chain of events:

1) **Gather all the sequences and pool them 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 `sample_seq1.fasta`, `sample_seq1_aln2.aln`, `sample_seq1_aln2.msf` and `sample_seq1_lib1.tc_lib`, plus the sequences initially gathered by **-infile**.

2) **Turn alignment(s) into libraries** (e.g. 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.

This procedure follows specific rules within T-Coffee; be carefull with the following rules:

- **Order**: the order in which sequences, methods, alignments and libraries are fed in is irrelevant.
- **Heterogeneity**: there is no need for each element (A, S, L) to contain the same sequences.
- **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.
- **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. 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.

```
aln 1:      hgab1 AAAAABAAAAA
aln 2:      hgab1 AAAAAAAAAACCC
consensus:  hgab1 AAAAABAAAAACCC
```

- **Substitution Matrices**: if the method is a substitution matrix (X) then no other type of information should be provided. This command results in a progressive alignment carried out on the sequences in seqfile. The procedure does not use any more the T-Coffee concistency based algorithm, but switches to a standard progressive alignment algorithm (like ClustalW or Pileup) much less accurate. In this context, appropriate gap penalties should be provided. The matrices are in the file `matrices.h` in the folder **"$HOME/tcoffee/Version_XX/src/"**. *Ad hoc* matrices can also be provided by the user (see the matrices format section at the end of this manual).

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

Warning: The matrix **X** does not have the same effect as using the **-matrix** flag, which defines the matrix that will be used while compiling the library while the Xmatrix defines the matrix used when assembling the final alignment.

- **Methods**: the method describer can either be built-in or be a file describing the method to be used (see chapter **T-Coffee Main Documentation, Internal/External Methods In T-Coffee** for more information). The exact syntax is provided later in this technical documentation.

6.1.2.2 Sequence input flags

- **-infile** (usage:**-infile=<filename>**) [cw]

Common multiple sequence alignments format constitute a valid input format. To remain compatible with ClustalW ([cw]) it is possible to indicate the sequences with this flag. T-Coffee automatically removes the gaps before doing the alignment, and this behaviour is different from that of ClustalW where the gaps are kept.

- **-get_type**

Forces T-Coffee to identify the sequences type (protein, DNA or RNA sequences).

- **-type** (usage:**-type=[DNA,RNA,PROTEIN]**) [cw]

This flag sets the type of the sequences. The. By default, it recognizes the sequence type. If omitted, the type is guessed automatically, but in case of low complexity or short sequences, it is recommended to set the type manually This flag is compatible with ClustalW.

- **-seq** (usage:**-seq=[<P,S><name>]**)

The flag **-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]**) [under evaluation]

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.The flag **-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. The flag **-seq_source=S** means that only sequences provided via the S tag will be considered. All the other sequences will be ignored. This flag was mostly designed for interactions between T-Coffee and T-CoffeeDPA (the large scale version of T-Coffee) which is now deprecated !!!

6.1.2.3 Other input flags (structure, tree, profile)

- **-pdb** (usage:**-pdb=<pdbid1>,<pdbid2>...**/[max 200])

It reads or fetch a PDB file or even to specify a chain or a sub-chain: PDBID(PDB_CHAIN)[opt] (FIRST,LAST)[opt]. It is also possible to input structures via the **-in** flag but in that case, you will need to use the TAG identifier (Ppdb1 Ppdb2...).

- **-usetree** (usage:**-usetree=<tree file>**) [cw]

This flag indicates that rather than computing a new dendrogram, T-Coffee must use a precomputed one in newick tree format (ClustalW Style). The tree files are in Phylip format and compatible with ClustalW. In most cases, using a precomputed tree will halve the computation time required by T-Coffee. It is also possible to use trees output by ClustalW, Phylip and some other tree generating software.

- **-profile** (usage:**-profile=[<name1>,<name2>,...]**/[max 200])

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. 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...).

```
Turning several MSA into a single profile:
$$: t_coffee -profile sample_seq1.aln,sample_seq1_aln2.aln -outfile=profile_aln

Using MSA as profiles:
$$: t_coffee -in Rsample_seq1.aln,Rsample_seq1_aln2.aln,Mslow_pair,Mlalign_id_pair \
    -outfile=profile_aln
```

- **-profile1/-profile2** (usage:**-profile1=[<prf1>]/-profile2=[<prf2>]**/one name only/) [cw]

It is similar to the previous command and was provided for compatibility with ClustalW.It accepts only one name in as parameter.

6.1.3 Output(s)

Those names, 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. In the T-Coffee output (displayed on screen), the output results appear in the following format (last lines displayed once the job is done):

```
OUTPUT RESULTS
##### File Type= <type> Format= <format> Name <filename>

File Type: can be GUIDE TREE, MSA, etc...
Format   : can be newick, html, aln, etc...
Name     : prefix is the name of the input, extension corresponds to the type &␣
↪format.
```

6.1.3.1 Output files, format & names

- **-run_name** (usage:**-run_name=<your run name>**)

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

- **-align** [cw]

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

- **-outfile** (usage:**-outfile=[out_aln,file,default,no]**)

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 the output of the resulting alignment; more than one format can be indicated. The supported input/output formats are listed below. The scoring output files rely mainly on the T-Coffee CORE index (you can find more information here).

```
Sequence format:
- clustalw_aln, clustalw
- gcg
- msf_aln
- pir_aln, pir_seq
- fasta_aln, fasta_seq
- phylip

Output format:
- 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
```

- **-seqnos** (usage:**-seqnos=[on,off]**/default:**-seqnos=off**)

Causes the output alignment to contain the number of residue at the end of each line.

6.1.3.2 Evaluation files

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 here. All these modes will be useful when generating colored version of the output, with the **-output** flag (cf. Chapter **T-Coffee Main Documentation, Section Preparing Your Data**).

- **-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. Several options are possible:

- **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.

```
$$: t_coffee sample_seq1.fasta -evaluate_mode t_coffee_slow -output score_ascii

$$: t_coffee sample_seq1.fasta -evaluate_mode t_coffee_fast -output score_ascii

$$: t_coffee sample_seq1.fasta -evaluate_mode t_coffee_non_extended -output score_
↪ascii
```

6.1.3.3 Alignments

- **-outseqweight** (usage:**-outseqweight=<filename>**/default:none)

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, refers to the **T-Coffee Main Documentation**.

- **-outorder** (usage:**-outorder=[input,aligned,filename]**/default:**-outorder=input**) [cw]

Sets the order of the sequences in the output alignment; by default, the sequences are kept in the original order of the input dataset. Using **-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. It is also possible to provide a file (a legal FASTA file) whose order will be used in the final MSA via **-outdorder=<filename>**.

- **-inorder** (usage:**-inorder=[input,aligned]**/default:**-inorder=aligned**) [cw]

MSAs based on dynamic programming depend slightly on the order in which the incoming sequences are provided. To prevent this effect sequences are arbitrarily sorted at the beginning of the program (**-inorder=aligned**). However, this affects the sequence order within the library. You can switch this off by stating **-inorder=input**.

- **-cpu** [Deprecated]

6.1.3.4 Librairies & trees

- **-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**

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

- **-newtree** (usage:**-newtree=<tree file>**)

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 Phylip.

Warning: Do NOT confuse this guide tree with a phylogenetic tree.

6.1.4 Alignment Computation

6.1.4.1 Library computation: methods and extension

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 tries to assemble the alignment with the highest level of consistency. You can think of the alignment as a list of constraints; the job of T-Coffee is to satisfy as many constraints as possible.

- **-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** [Unsupported]
- **-align_pdb_hasch_mode** [Unsupported]
- **-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=[see below...]**/default:**-extend=very_fast_triplet**)

Development only. Controls the algorithm for matrix extension. Available SUPPORTED modes include: **fast_triplet**, **very_fast_triplet** (limited to the **-max_n_pair** best sequence pairs when aligning two profiles), **slow_triplet** (exhaustive use of all the triplets), **matrix** (use of the matrix **-matrix**) and **fast_matrix** (use of the matrix **-matrix**). The following are NOT SUPPORTED: relative_triplet, g_coffee, g_coffee_quadruplets, mixt, quadruplet, test. 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**).

- **-weight** (usage:**-weight=[winsimN,sim,sim_<matrix_name,matrix_file>,integer] /** default:**-weight=sim**)

Weight defines the way alignments are weighted when turned into a library. Overweighting can be obtained with the OW<X> weight mode; 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. However, in our hands, this type of weighting scheme has not provided any significant improvement over the standard sim value (the 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 prespecified score (for instance if they come from a structure superimposition program).

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

$$: t_coffee sample_seq1.fasta -weight=1000 -out_lib=test.tc_lib

$$: t_coffee sample_seq1.fasta -weight=sim_pam250mt -out_lib=test.tc_lib
```

Several options are available:

- **sim** : indicates that the weight equals the average identity within the sequences containing the matched residues.
- **OW<X>** : will cause the sim weight to be multiplied by X.
- **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 (Refer to the next section about matrices). For instance, -weight=sim_pam250mt indicates that the grouping used for similarity will be the set of classes with positive substitutions.
- **sim_clustalw_col**: categories of clustalw marked with ":".
- **sim_clustalw_dot**: categories of clustalw marked with ".".
- **-lib_list** (usage:**-lib_list=<filename>**) [Unsupported]

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

- **-do_self** [Unsupported]

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** [Unsupported]
- **-compact** [Unsupported]
- **-clean** [Unsupported]
- **-maximise** [Unsupported]

6.1.4.2 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-like)
- aln       : read the distances on a precomputed MSA
```

- **-quicktree** [cw]

Causes T-Coffee to compute a fast approximate guide tree; this flag is kept for compatibility with ClustalW.

```
$$: t_coffee sample_seq1.fasta -distance_matrix_mode=very_fast

$$: t_coffee sample_seq1.fasta -quicktree
```

6.1.4.3 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 unweighted library; weights can be output using the **-outseqweight** flag; you can use your own weights (see **T-Coffee Parameter Files Format** subsection).

6.1.4.4 Pairwise 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 **-ktup**
3) Dynamic programming runs on the **-ndiag** best diagonals whose score is higher than **-diag_threshold**, the way diagonals are scored is controlled via **-diag_mode**.
4) The Dynamic computation is made to optimize either the library scoring scheme (as defined by the **-in**) or a substitution matrix as provided via **-matrix**. The penalty scheme is defined by **-gapopen** and **-gapext**. If **-gapopen** is undefined, the value defined via **-cosmetic_penalty** is used instead.
5) Terminal gaps are scored according to **-tg_mode**.

- **-dp_mode** (usage:**-dp_mode=<string>**/default:**-dp_mode=cfasta_fair_wise**).

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

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

The possible modes are:

- **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= 1 or 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 but 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**.
- **-ktuple** (usage:**-ktuple=[value]**/default:**-ktuple=1 or 2**).

Indicates the ktuple size for cfasta_pair_wise and fasta_pair_wise of **-dp_mode**. It is set to 1 for proteins, and 2 for DNA. The alphabet used for protein can be a degenerated version, set with **-sim_matrix**.

- **-ndiag** (usage:**-ndiag=[value]**/default:**-ndiag=0**)

Indicates the number of diagonals used by the fasta_pair_wise algorithm (**-dp_mode**). When **-ndiag=0**, n_diag=Log (length of the smallest sequence)+1. When **-ndiag & -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, and "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. A value of 0: indicates that **-ndiag** (seen before) should be used to select the diagonals.

- **-sim_matrix** (usage:**-sim_matrix=[string]**/default:**-sim_matrix=vasiliky**)

Indicates the manner in which the aminoacid alphabet is degenerated when hashing in the fasta_pairwise dynamic programming. Standard ClustalW matrices are all valid. They are used to define groups of aminoacids 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], and all other residues kept alone. To keep the standard alphabet non degenerated, better use **-sim_matrix=idmat**.

- **-matrix** (usage:**-matrix=[blosum62mt,...]**/default:**-matrix=blosum62mt**) [cw]

  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...). Users can also provide their own matrices, using your own *matrix <http://www.tcoffee.org/Projects/tcoffee/documentation/index.html#blast-format-recommended>*.

- **-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 not 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. The penalty must be negative. If no value is provided when using a substitution matrix, a value will be automatically computed.

Hint: Here are some guidelines regarding the tuning of **-gapopen** and **-gapext**. In T-Coffee matches get a score between 0 (match) and 1000 (match perfectly consistent with the library). The default cosmetic penalty is set to -50 (5% of a perfect match). If you want to tune **-gaopen** and see a strong effect, you should therefore consider values between 0 and -1000.

- **-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 (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**)

The values indicate a penalty on the terminal gaps: "0" a penalty of -gapopen + -gapext*len; "1" a penalty of -gapext*len; and "2" for no penalty associated to terminal gaps.

- **-fgapopen** [Unsupported]
- **-fgapext** [Unsupported]

6.1.4.5 Multiple alignment computation

- **-one2all** (usage:**-one2all=[name]**)

Will generate a one to all library with respect to the specified sequence and will then align all the sequences in turn to that sequence, in a sequence determined by the order in which the sequences were provided.If **-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** (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 [Unsupported]).

2) When MSAs are provided via **-profile, -profile1 or -profile2**. In these situations, the **-profile_mode** value influences the alignment computation, these values are: a) **-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; b) **-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.

- **-msa_mode** (usage:**-msa_mode=[tree,graph,precomputed]**/default:**-evaluate_mode=tree**) [Unsupported]

6.1.4.6 Large scale aligment computation [Unsupported]

- **-dpa** (usage:**-dpa**/default:**unset**) [Unsupported]

This flag triggers the dpa mode. This mode involves computing the <-dpa_tree> that is then resolved into an alignment on buckets of <-dpa_nseq> sequences using <-dpa_method> as an aligner. If no <-dpa_method> is provided the T-Coffee aligner is used and all the parameters (including modes) are passed to T-Coffee. Note that all the output parameters are not supported. Note tha sequences should be input using the -seq flag.

- **-dpa_nseq** (usage:**-dpa_nseq=[ineger]**/default:**-dpa_nseq=30**)

  Specifies the maximum bucket size when using the dpa mode. The buckets are provided to the <dpa_method> or to the remainder of the command line

- **-dpa_tree** (usage:**-dpa_tree=[file, method]**/default:**-dpa_tree=kmtree**)

  Specifies the dpa tree

```
dpa_tree modes
##: catswl       --- caterpilar tree with sequences ordered according to their␣
↪distance from the average swl vector
##:                 the swl vector of a sequence is defined as the SMith and␣
↪Waterman Length against the -swlN N sequences
##: catlong      --- caterpilar tree with sequences ordered according to their length␣
↪(longuest on root)
##: catshort     --- caterpilar tree with sequences ordered according to their length␣
↪(shortest on root)
##: swldnd       --- Kmeans ran on the <-swlN> dimention vector where each component␣
↪is the SW length of a sequence against a swlN seed
##: kmdnd        --- use kmeans based on triaa vectors (fast)
##: codnd        --- clustalo guide tree, obtainned by running clustalo externaly
##: cwdnd        --- clustalw guide tree, obtainned by running clustalw externaly
##: #<pg>        --- runs pg <seq> > stdout that outputs the tree on the stdou
##: parttree     --- MAFFT parttree (external)
##: dpparttree   --- MAFFT dpparttree (external)
##: fastparttree --- MAFFT fastparttree (external)
```

- **-dpa_swlN** (usage:**-dpa_swlN=[integer]**/default:**-dpa_swlN=10**)

  When computing sw vetcors, specifies the number of N seed sequences to do an all-against-N. The sequences are selected evenly among the full sequences set sorted by size.

- **-dpa_weight** (usage:**-dpa_weight=[file, method]**/default:**-dpa_weight=longuest**). Weight used to push sequences from buckets to buckets. Sequence with the highest weight gets pushed one lebel up.

```
dpa_weight modes
##: longuest     --- push the longuest sequence
##: shortest     --- push the shortest sequence
##: name         --- push the sequence with the shorttes name (debug purpose)
##: #<pg>        --- runs pg <seq> > stdout that outputs a two column file <seqname>
↪<float weight value> (fasta w/o sequences can be read)
##: <file>       --- reads weights from two column file <seqname> <float weight value>
↪ (fasta w/o sequences can be read)
##: kmeans       --- assign to each sequence the size of its kmeans cluster, as␣
↪computed using triaa vectors and requesting 100 groups
##: swa,iswa     --- assigns to each sequence the average length of its SW alignment␣
↪(matched residues) against the -dpa_swlN seed sequences, iswa for invert
##: swl,iswl     --- assign each sequence a weight equal to its distance from the␣
↪average sequence using a swlN component vector, iswl for invert
##: diaa,idiaa   --- assign each sequence a weight equal to its distance from the␣
↪average sequence using a diaa component vector, idiaa fro invert
##: triaa,itriaa --- assign each sequence a weight equal to its distance from the␣
↪average sequence using a triaa component vector, itriaa fro invert
```

6.1.4.7 Local alignments computation [Unsupported]

It is possible to compute multiple local alignments, using the moca routine. MOCCA 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 the appropriate parameters.

- **-domain/-mocca** (usage:**-domain**) [Unsupported]

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

```
$#: t_coffee -in Ssample_seq1.fasta,Mlalign_rs_s_pair -out_lib=sample_lib1.mocca_lib \
    -domain -start=100 -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:

```
$#: t_coffee -in sample_lib1.mocca_lib -domain -start=100 -len=60
```

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

- **-start** (usage:**-start=[integer]**) [Unsupported]

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=[integer]**) [Unsupported]

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

- **-scale** (usage:**-scale=[integer]**/default:**-scale=-100**) [Unsupported]

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** [Unsupported]

Launches an interactive MOCCA session.

6.1.4.8 Alignment post-processing

- **-clean_aln**

This flag causes T-Coffee to post-process the MSA. 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 constitency residues and works its way left to right, top to bottom along the alignment. You can require this operation to be carried out for several cycles using the **-clean_iterations** flag. The rationale behind this operation is mostly cosmetic. In order to ensure a decent looking alignment, the GOP is set to -20 and the GEP to -1. There is no penalty for terminal gaps, and the matrix is blosum62mt. Gaps are always considered to have a reliability score of 0. 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=[1-X]**/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**.

6.1.5 Template based modes

6.1.5.1 Database searches parameters

These parameters are used when running template based modes of T-Coffee such as EXPRESSO (3D), PSI-Cofee (homology extension), TM/PSI-Coffee (homology extension/reduced databases), or accurate (mixture of profile and 3D templates).

- **-blast_server** (sage:**-blast_server=[EBI,NCBI,LOCAL]**/default:**-blast_server=EBI**)

Defines which way BLAST will be used, either through web services or locally. To have more information about BLAST, refer to the **T-Coffee Installation** chapter.

- **-psiJ** (usage:**-psiJ= <D>**/default:1)

Defines the number of psi-BLAST iterations

- **-protein_db** (usage:**-protein_db=<database>**/default:nr database)

Database used for the construction of a PSI-BLAST profile.

- **-prot_min_sim** (usage:**-prot_min_sim= [%identity]**/default:40)

Minimum identity for inclusion of a sequence in a PSI-BLAST profile.

- **-prot_max_sim** (usage:**-prot_max_sim= [%identity]**/default:90)

Maximum identity for inclusion of a sequence in a PSI-BLAST profile.

- **-prot_min_cov** (usage:**-prot_min_cov=[0-100]**/default:40)

Minimum coverage for inclusion of a sequence in a PSI-BLAST profile.

- **-pdb_db** (usage:**-protein_db=[database]**/Default:PDB database)

Database for PDB template to be selected by EXPRESSO. By default it runs on the PDB, but you can specify a version installed locally on your system,

- **-pdb_type** (usage:**-pdb_type=[d,n,m,dnm,dn]**/default:**-pdb_type=d**)

The different types are as follow: “d” stands for XRAY structures (diffraction), “n” stands for NMR structures, “e” stands for EM structures and “m” stands for models.

- **-pdb_min_sim** (usage:**-pdb_min_sim= [%identity]**/default:35)

Minimum % identity for a PDB template to be selected by Expresso.

- **-pdb_max_sim** (usage:**-pdb_max_sim= <%identity>**/default:100)

Maximum % identity for a PDB template to be selected by Expresso.

- **-pdb_min_cov** (usage:**-pdb_min_cov= <0-100t>**/default:50)

Minimum coverage for a PDB template to be selected by Expresso. PSI-Coffee and TM-Coffee Parameters

6.1.5.2 PSI-Coffee and TM-Coffee parameters

Profiles returned by Blast and PSI-Blast are usualy too large and very redundant. They are trimmed to the <-psitrim=40> most informative sequences, as determined by applying the regressive algorithm onto the returned sequences using <psitrim_tree=codnd>

6.1.5.3 Using structures

- **-mode** (usage:**-mode=[3dcoffee,expresso]**)

Refer to the **T-Coffee Main Documentation** for the structural modes of T-Coffee.

- **-check_pdb_status**

Forces T-Coffee to run **extract_from_pdb** to check the PDB status of each sequence. This can considerably slow down the program.

6.1.5.4 Using/finding PDB templates for the sequences

- **-struc_to_use** (usage:**-struc_to_use=[struc1, struc2...]**/ Default:none)

Restricts the 3D-Coffee to a set of predefined structures.

- **-template_file** (usage:**-template_file =[<filename>,<SCRIPT_scriptname>,SELF_TAG_,SEQFILE_TAG_<filename>,PDB,**

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. Each template will be used in place of the sequence with the appropriate method. There are several ways to pass the templates, the format of the template file being as followed "<sequence name> <TAG> <template name>":

Using a "filename":

You provide to T-Coffee an existing template file either created by Expresso or on your own. Different types of templates exist but all with the similar format and rules; the type of template on which a method works is declared with the SEQ_TYPE parameter in the method configuration file.

```
 Template file format:
 ><sequence name> _P_ <PDB template>
 ><sequence name> _G_ <gene template>
 ><sequence name> _R_ <MSA template>
 ><sequence name> _F_ <RNA Secondary Structure>
 ><sequence name> _T_ <Transmembrane Secondary Structure>
 ><sequence name> _E_ <Protein Secondary Structure>

 Template file rules:
 - 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

 Template based method types:
- 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)
```

Using "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 (See box 1.). It is also possible to pass some parameters, use @ as a separator and # in place of the = sign (See box 2.). For instance, if you want to call the a script named **blast.pl** with the following parameters. Bear in mind that the input/output flags will then be concatenated to this command line so that T-Coffee ends up calling the program using the following system call.

```
1) Running using a script:
##: scriptname -infile=<your sequences> -outfile=<template_file>

2) Running blast.pl:
##: blast.pl -db=pdb -dir=/local/test
##: SCRIPT_blast.pl@db#pdb@dir#/local/test
##: blast.pl -db=pdb -dir=/local/test -infile=<some tmp file> -outfile=<another tmp␣
↪file>
```

Using "SELF_TAG":

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.

```
$$: t_coffee three_pdb.fasta -template_file SELF_P_
```

Using "SEQFILE_TAG_filename"

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`.

6.1.5.5 Using structure for MSA evaluation

MSA can be evaluated using structures with T-Coffee special mode APDB/iRMSD (see **T-Coffee Main Documentation**). The requirements to run these modes are similar to running a structure based MSA with Expresso/3D-Coffee. Here we describe the different parameters associated with the APDB mode.

- **-n_excluded_nb** (usage:**-n_excluded_nb=[integer]**/default:1)

When evaluating the local score of a pair of aligned residues, the residues immediately next to that column should not contribute to the measure. By default the first to the left and first to the right are excluded.

- **-maximum_distance** (usage:**-maximum_distance=[float]**/default:10)

Size of the neighborhood considered around every residue. If .-local_mode is set to sphere, -maximum_distance is the radius of a sphere centered around each residue. If -local_mode is set to window, then -maximum_distance is the size of the half window (i.e. window_size=-maximum_distance*2+1).

- **-similarity_threshold** (usage:**-similarity_threshold=[integer]**/default:70)

Fraction of the neighborhood that must be supportive for a pair of residue to be considered correct in APDB. The neighborhood is a sphere defined by -maximum_distance, and the support is defined by **-md_threshold**.

- **-local_mode** (usage:**-local_mode=[sphere,window]**/default:sphere)

Defines the shape of a neighborhood, either as a sphere or as a window.

- **-filter** (usage:**-filter=<0.00-1.00>**/default:1.00)

Defines the centiles that should be kept when making the local measure. Foir instance, -filter=0.90 means that the the 10 last centiles will be removed from the evaluation. The filtration is carried out on the iRMSD values.

- **print_rapdb** (usage:**-print_rapdb=[FLAG]**/default:off) [Unsupported]

This causes the prints out of the exact neighborhood of every considered pair of residues.

- **color_mode** (usage:**-color_mode=[apdb,irmsd]**/default:apdb)

This flag is meant to control the colored APDB output (local score). This file will either display the local APDB score or the local iRSMD.

6.2 T-Coffee Parameter Files Format

6.2.1 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:

1) **No space**: 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). For instance, sequences with spaces ">seq1 human_myc" will be turned into ">seq1".

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.

6.2.2 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.

- **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.
- **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.
- **RNA Structures**: RNA structures can either be coded as T-Coffee libraries, with each line indicating two paired residues, or as alifold output. The selex format is also partly supported (see the seq_reformat tutorial on RNA sequences handling).
- **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
```

(continues on next page)

(continued from previous page)

```
<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.

6.2.3 Libraries

6.2.3.1 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 that here is a space between the ! And SEQ_1_TO_N...and 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:

```
#1 2
12 13 99
```

(continues on next page)

(continued from previous page)

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

It is also possible to declare ranges of residues rather than single pairs. For instance, the following:

```
#0 1
+BLOCK+ 10 12 14 99
+BLOCK+ 15 30 40 99
#0 2
15 16 99
#0 1
12 14 70
```

The first statement BLOCK declares a BLOCK of length 10, that starts on position 12 of sequence 1 and position 14 of sequence 2 and where each pair of residues within the block has a score of 99. The second BLOCK starts on residue 30 of 1, residue 40 of 2 and extends for 15 residues. Blocks can overalp and be incompatible with one another, just like single constraints.

6.2.3.2 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)
=> N 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.

6.2.3.3 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
...
```

6.2.4 Substitution matrices

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

6.2.4.1 BLAST format [Recommended]

The alphabet can be freely defined

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

6.2.4.2 ClustalW style [Deprecated]

```
# CLUSTALW_MATRIX FORMAT
$
v1
v2 v3
v4 v5 v6
...
$
```

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

6.2.5 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. Unweighted sequences will see their weight set to 1.

6.2.6 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>=.....
```

6.3 Technical Notes

These notes are only meant for internal development.

6.3.1 Building a T-Coffee server

We maintain a T-Coffee server (www.tcoffee.org). We will be pleased to provide anyone who wants to set up a similar service with the sources

6.3.1.1 Environment Variables

T-Coffee stores a lots of information in locations that may be unsuitable when running a server.

By default, T-Coffee will generate and rely on the following directory structure:

```
Directory structure:
##: /home/your account/ #HOME_4_TCOFFEE
##: HOME_4_TCOFFEE/.t_coffee/  #DIR_4_TCOFFEE
##: DIR_4_TCOFFEE/cache #CACHE_4_TCOFFEE
##: DIR_4_TCOFFEE/tmp #TMP_4_TCOFFEE
##: DIR_4_TCOFFEE/methods #METHOS_4_TCOFFEE
##: DIR_4_TCOFFEE/mcoffee #MCOFFEE_4_TCOFFEE
```

By default, all these directories are automatically created, following the dependencies suggested here. The first step is the determination of the HOME. By default the program tries to use HOME_4_TCOFFEE, then the HOME variable and TMP or TEMP if HOME is not set on your system or your account. It is your responsibility to make sure that one of these variables is set to some valid location where the T-Coffee process is allowed to read and write. If no valid location can be found for HOME_4_TCOFFEE, the program exits. If you are running T-Coffee on a server, we recommend to hard set the following locations, where your scratch is a valid location.

```
T-Coffee server configuration:

##: HOME_4_TCOFFEE='your scratch'
##: TMP_4_TCOFFEE='your scratch'
##: DIR_4_TCOFFEE='your scratch'
##: CACHE_4_TCOFFEE='your scratch'
##: NO_ERROR_REPORT_4_TCOFFEE=1
```

Note that it is a good idea to have a cron job that cleans up this scratch area once in a while.

6.3.1.2 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 (refers to the **T-Coffee Parameters & Flags**, in the **Output** subsection about trees). Choose the name so that two processes may not over-write each other dnd file.

6.3.1.3 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.

6.3.1.4 Other Programs

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

6.3.2 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 all * 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.

6.3.3 Development

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

- **profile to list**

prf1: profile containing one structure prf2: profile containing one structure

```
$$: t_coffee -profile sample_profile1.aln,sample_profile2.aln -mode=3dcoffee
```

- **command line list**

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

1) External methods:

```
$$: t_coffee sample_seq1.fasta -in=Mclustalw_pair,Mclustalw_msa,Mslow_pair -
↪outfile=text
```

2) List of command lines provided by James Watson to crash T-Coffee before version 3.40:

```
$$: t_coffee -mode 3dcoffee -in Sthree_pdb.fasta P1PPG
$$: t_coffee -mode 3dcoffee -in Sthree_pdb.fasta -template_file SELF_P_
```

3) Command line to read 'relaxed' PDB files...

```
$$: t_coffee -in Msap_pair Ssample_3Dseq1.fasta -template_file sample_3Dseq1.template␣
↪\
    -weight 1000 -out_lib sample_3Dseq1.tc_lib -lib_only
```

4) Parsing long sequence lines:

```
$$: t_coffee -in Aproteases_large.aln -outfile test.aln
```

6.4 T-Coffee Test

Before the realease of a new T-Coffee version, many command lines in the documentation are programmatically tested. The purpose of this document is to explain which command lines are tested, how new command lines can be added to the documentation and how to perform a documentation check before a new release.

6.4.1 Introducing a new command to test

In this documentation you will find three types of command lines some of which are up and running on the examples files. Whenever adding a new command, input files must be added to the repository directory ./examples/. New commands can be also be built using the existing files, or they can depend on files newly added to the repository. Here are the three different flag types of command lines you will fine in the documentation "$$", "$#" and "##":

- **$$: t_coffee...**

Corresponds to all command that will be tested systematically in a programmatic manner. Other command lines starting with different symbols are not checked.

- **$#: t_coffee...**

Corresponds to a command that could be tested but will not be, either for the sake of time or because it is currently unstable. When a new release needs to be urgently made available because of a critical fix, it is advisable to comment out this way non critical command lines failing the test.

- **##: t_coffee...**

These commands are never tested, either because they contain system dependant information or non programmatic information.

6.4.2 Checking the documentation

Our procedure uses **doc2test.pl** to extract all command lines from the .rst in docs and will run these command lines. When a reference output is available in /testsuite/docs/ref/ it will produce an other output and compare it with the reference. The new output will be put in /testsuite/docs/latest/. The script will eventually produce a global report that is to be found in /testsuite/docs/log/validation.log. The output of the failed scripts will be put in /testsuite/docs/failed. All the reference output will be added to git repository. The simplest way to check the documentation is to run the following command:

```
Check the documentation:
##: ./lib/perl/lib/perl4makefile/doc2test.pl -mode update
```

When changes are made in the list of command line, we use the following mode whichwill only run the new command lines and report (defined as those not having a reference output in /testsuite/docs/ref/). This mode is the default mode:

```
Check new command lines:
##: ./lib/perl/lib/perl4makefile/doc2test.pl -mode new
```

This mode will only run the command lines that have previously failed, as indicated by the presence of an output in the failed directory.

```
Check failed command lines:
##: ./lib/perl/lib/perl4makefile/doc2test.pl -mode failed
```

6.4.3 Compiling the documentation

The documentation can be compiled using the following command; it will cause the test to stop whenever a failed is encountered.

```
Compile the documentation:
##: ./lib/perl/lib/perl4makefile/doc2test.pl -mode update -stop_on_failed
```

To have a clean start, you can also reset all files. This command will cause all the reference files to be erased:

```
Reset reference files:
##: ./lib/perl/lib/perl4makefile/doc2test.pl -mode reset
```

To removed unnecessary files, the following command will cause all the files in ./examples/ that are not used for any /docs/*.rst commands to be deleted. They will be deleted from both their current locations and the git repository. The remaining files will be added to the git repository.

```
Remove unnecessary files:
##: ./lib/perl/lib/perl4makefile/doc2test.pl -clean
```

6.4.4 Making a new release

The Beta and Stable releases can be done with the makefile, from the t_coffee/src/ directory

```
1-git commit -a -m 'what you did'
2-make beta_release OR make stable_release -- This will populate the source␣
↪directories
3-git push
```

CHAPTER 7

T-Coffee Web Server

Warning: This chapter is currently under maintenance. Its aim is to describe what the T-Coffee web server is doing programmatically when you are using it; it provides technical details about the restrictions and commands in use.

In this chapter we describe briefly the available T-Coffee web servers and their usage (command lines and examples files); if you need more advanced options, you should use the T-Coffee package via command lines. The web server is a reduced version of the T-Coffee package, containing all T-Coffee modes for aligning protein, RNA or DNA sequences; it also contains the evaluation and downstream analysis tools (TCS, iRMSD/APDB, STRIKE, T-RMSD). Currently, all reformatting utilities are not available on the web server, however, you can choose some reformatting options related to the ouput format. Here we present briefly the webserver and the T-Coffee commands it contains.

7.1 General

Most of the following command lines used by the web server contains lots of different options called with flags; here is a summary of the options common to all comand lines:

- **-in**: your input file
- **-output**: specifies the output files you require
- **-maxnseq**: maximum number of sequences you can run (variable depending on the mode)
- **-maxlength**: maximum length of your sequences (variable depending on the mode)
- **-case=upper**: all residues/nucleotids will be upper case
- **-seqnos=off**: the size of the sequences is not indicated on the MSA
- **-outorder=input**: orders the sequences in the final MSA as in the input dataset
- **-run_name**: name of the job on the cluster
- **-multi_core=4**: uses only 4 cores on your server when running the job

- **-quiet=stdout**: standard verbose output

7.2 T-Coffee Simple MSA

The central part of the web server is the T-Coffee aligner, you can use it to align any kind of sequences (Protein, RNA or DNA alike). The command it runs is the following:

```
$#: t_coffee -in=data_93c5fbb0.in -mode=regular -output=score_html clustalw_aln fasta_
↪aln \
    score_ascii phylip -maxnseq=150 -maxlen=10000 -case=upper -seqnos=off -
↪outorder=input \
    -run_name=result -multi_core=4 -quiet=stdout
```

7.3 Protein Sequences

7.3.1 Expresso (using 3D structures)

```
$#: t_coffee -in=data_93c5fbb0.in -mode=expresso -blast=LOCAL -pdb_db=/db/pdb/derived_
↪data_\
    format/blast/2016-01-01/pdb_seqres.fa -evaluate_mode=t_coffee_slow -output=score_
↪html \
    clustalw_aln fasta_aln score_ascii phylip -maxnseq=150 -maxlen=2500 -case=upper -
↪seqnos= \
    off -outorder=input -run_name=result -multi_core=4 -quiet=stdout
```

7.3.2 M-Coffee (combining multiple methods)

```
$#: t_coffee -in=data_93c5fbb0.in  Mpcma_msa Mmafft_msa Mclustalw_msa Mdialigntx_msa␣
↪Mpoa_msa \
    Mmuscle_msa Mprobcons_msa Mt_coffee_msa -output=score_html clustalw_aln fasta_aln␣
↪\
    score_ascii phylip -tree -maxnseq=150 -maxlen=2500 -case=upper -seqnos=off -
↪outorder=input \
    -run_name=result -multi_core=4 -quiet=stdout
```

7.3.3 PSI/TM-Coffee (transmembrane proteins)

Two options are available (in addition to the choice of the database): without transmembrane prediction or with prediction (displayed color coded on the html file).

```
$#: tmcoffee.sh -in data_9df741d4.in -mode psicoffee -blast_server LOCAL --search-db
↪'UniRef50 \
    -- Very Fast/Rough' --search-type '' -prot_min_sim 50 -prot_max_sim 90 -prot_min_
↪cov 70 \
    --search-out 'clustalw_aln fasta_aln score_ascii phylip score_html' -maxnseq 1000␣
↪-maxlen \
    =5000 -case upper -seqnos=off -outorder input -run_name result -multi_core 4 -
↪quiet=stdout
```

(continues on next page)

(continued from previous page)

```
$#: tmcoffee.sh -in data_9df741d4.in -mode psicoffee -blast_server LOCAL --search-db
↪'UniRef50 \
    -- Very Fast/Rough' --search-type 'transmembrane' -prot_min_sim 50 -prot_max_sim␣
↪90 \
    -prot_min_cov 70 --search-out 'clustalw_aln fasta_aln score_ascii phylip score_
↪html'
    -maxnseq 1000 -maxlen 5000 -case upper -seqnos off -outorder input -run_name␣
↪result °
    -multi_core 4 -quiet=stdout
```

7.3.4 PSI-Coffee (homology extension)

```
$#: t_coffee -in=data_93c5fbb0.in -mode=psicoffee -blast=LOCAL -protein_db=/db/ncbi/
↪201511/ \
    blast/db/nr.fa -output=score_html clustalw_aln fasta_aln score_ascii phylip -
↪maxnseq=150 \
    -maxlen=2500 -case=upper -seqnos=off -outorder=input -run_name=result -multi_
↪core=4 \
    -quiet=stdout
```

7.4 RNA Sequences

7.4.1 R-Coffee (using 2D prediction)

```
$#: t_coffee -in=data_29091222.in -method=mafft_msa muscle_msa probconsRNA_msa -
↪output= \
    score_html clustalw_aln fasta_aln score_ascii phylip -maxnseq=150 -maxlen=2500 -
↪case=upper \
    -seqnos=off -outorder=input -run_name=result -tree -special_mode=rcoffee -method_
↪limits= \
    consan_pair 5 150 -multi_core=4 -quiet=stdout
```

7.4.2 SARA-Coffee (using 3D structures)

SARA-Coffee is a bit complicated to run, it uses several third party packages and is run through a script and environment variables we set up; here is what it looks like:

```
##: export X3DNA=/data/www-cn/sara_coffee_package/X3DNA;
##: export PDB_DIR=/data/www-cn/sara_coffee_package/PDBdir/;
##: export NO_REMOTE_PDB_DIR=1;
##: unset MAFFT_BINARIES;
##: cd $CACHE_4_TCOFFEE
##: ln -s /data/www-cn/sara_coffee_package/pdb_entry_type.txt);
$#: t_coffee -in data_3e6e7aec.in -method sara_pair -template_file \
    /data/www-cn/sara_coffee_package/TEMPLATEFILE,RNA -extend_mode rna2 -relax_lib 0 -
↪transform \
    dna2rna -run_name=result -output score_html clustalw_aln -case=upper -seqnos=off -
↪outorder= \
    input -multi_core=4 -pdb_min_sim 0 -quiet stdout
```

7.4.3 RM-Coffee (combining multiple methods)

Not yet available...

7.5 DNA Sequences

7.5.1 M-Coffee (combining multiple methods)

For now, M-Coffee by default is the same for DNA, RNA and protein sequences alike. There is no specific M-Coffee for DNA sequences.

7.5.2 Pro-Coffee (homologous promoter regions)

```
$#: t_coffee -in=data_476efe5f.in -mode=procoffee -output=score_html clustalw_aln␣
↪fasta_aln \
    score_ascii phylip -maxnseq=150 -maxlen=10000 -case=upper -seqnos=off -
↪outorder=input \
    -run_name=result -multi_core=4 -quiet=stdout
```

7.6 Evaluation Tools

7.6.1 TCS (Transitive Consistency Score)

```
##: tcs.sh -infile data_a98d61a6.in -in Mproba_pair -score 1 -output clustalw_aln␣
↪fasta_aln \
    phylip score_ascii tcs_weighted tcs_replicate score_html -maxnseq 1000 -maxlen␣
↪8000 \
    -seqnos=off -run_name result -multi_core 4 --filter-type column --filter-min 4 --
↪filter-max \
    9 --filter-gap yes -quiet=stdout
```

7.6.2 iRMSD/APDB (MSA structural evaluation) (under maintenance...)

```
$#: t_coffee -other_pg apdb -aln data_c7151320.in -apdb_outfile default -outfile␣
↪default \
    -io_format hsg3 -output score_html -maximum_distance 10 -md_threshold 2.0 -
↪similarity_ \
    threshold 70 -template_file EXPRESSO -run_name result -quiet stdout
```

7.6.3 T-RMSD (structural clustering)

```
$#: t_coffee -in=data_b89d3438.in -mode=expresso -cache=$PWD -blast=LOCAL -pdb_db=/db/
↪pdb/ \
    derived_data_format/blast/2016-01-01/pdb_seqres.fa -evaluate_mode=t_coffee_slow -
↪output= \
    aln score_html -maxnseq=150 -maxlen=2500 -case=upper -outorder=input -run_
↪name=result \
```

(continues on next page)

(continued from previous page)

```
    -multi_core=4 -quiet=stdout; t_coffee -other_pg trmsd result.aln -template_file \
    result_pdb1.template_list -output color_html 2>&1; [ -e result.struc_tree.
↪consensus ]
```

7.6.4 STRIKE (MSA evaluation with single structure)

:: $#: wget ftp://ftp.rcsb.org/pub/pdb/derived_data/pdb_seqres.txt $#: install ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST

$#: t_coffee -other_pg strike <sequence file> -template_file PDB -pdb_db <pdb_seqres> -blast_server LOCAL

CHAPTER 8

FAQ

8.1 Abnormal Terminations & Results

8.1.1 Q: The program keeps crashing when I give my sequences

A-1: This may be a format problem. Try to reformat your sequences, we recommend the FASTA format. If the problem persists, contact us.

A-2: 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 specify the sequence format with **-type** and the type of sequence **DNA, RNA or PROTEIN**.

A-3: Costly computation or data gathered over the net is stored by T-Coffee in a cache directory. Sometimes, some of these files can be corrupted and cause an abnormal termination. You can either empty the cache (~/.t_coffee/cache/), request T-Coffee to run without using it (command 1), or update the files corresponding to your data (command 2).

```
Command 1: No cache
$$: t_coffee -pdb=struc1.pdb,struc2.pdb,struc3.pdb -method sap_pair -cache=no

Command 2: Update cache
$$: t_coffee -pdb=struc1.pdb,struc2.pdb,struc3.pdb -method sap_pair -cache=update
```

8.1.2 Q: The default alignment is not good enough

A: see next question

8.1.3 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.

8.1.4 Q: The program is crashing

A: If you get the message **FAILED TO ALLOCATE REQUIRED MEMORY** refer to the next question. Otherwise, if the program crashes check whether you are using the right syntax. If the problem persists, contact us.

8.1.5 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 (command 1). 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 (command 2). If you keep running out of memory, you may also want to reduce the number of sequences using **-maxnseq**.

```
Command 1:
$$: t_coffee sample_seq1.fasta -mode low_memory

Command 2:
$$: t_coffee sample_seq1.fasta -method=slow_pair,lalign_id_pair -distance_matrix_mode␣
↪\
    =idscore -dp_mode=myers_miller_pair_wise
```

8.2 Input/Output control

8.2.1 Q: How many sequences can T_Coffee handle?

A: T-Coffee is recommanded for a maximum of 200. Above this number, the regreessive flavor (**t_coffee -reg**) should be used

8.2.2 Q: Can I change line length in the defaut format?

A: Yes, by using the flag **-aln_line_length**.

8.2.3 Q: Can I prevent the output of all the warnings?

A: Yes, by setting **-no_warning**.

8.2.4 Q: How many ways to pass parameters to t_coffee?

A: Refer to the **T-Coffee Technical Documentation**, Section **T-Coffee Parameters & Flags**.

8.2.5 Q: How can I change the default output format?

A: See the **-output** option in the **T-Coffee Main Documentation**; nearly all common output formats are recognized by T-Coffee.

8.2.6 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**).

8.2.7 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 this instruction will output the tree (in Newick format) and the alignment to stdout.

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

8.2.8 Q: Is it possible to pipe stuff into T_Coffee?

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

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

is equivalent to:

$$: t_coffee sample_seq1.fasta
```

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

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

or

$$: echo -seq=sample_seq1.fasta -method=clustalw_pair | t_coffee -parameters=stdin
```

8.2.9 Q: Can I read my parameters from a file?

A: See the **T-Coffee Technical Documentation**.

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

A: Use the **-run_name** flag:

```
$$: t_coffee sample_seq1.fasta -run_name=luke_skywalker
```

8.2.11 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:

```
$$: t_coffee sample_aln1.aln -in proba_pair

$$: t_coffee -seq=sample_aln1.aln -method=slow_pair,lalign_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.

8.2.12 Q: I only want to produce a library

A: use the **-lib_only** flag; but note that this supersedes the use of the **-convert** flag. Its main advantage is to restrict computation time to the actual library computation.

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

8.2.13 Q: I want to turn an alignment into a library

A: use the **-lib_only** flag (command 1). It is also possible to control the weight associated with this alignment with the flag **-weight** (command 2).

```
Command 1:
$$: t_coffee -in=Asample_aln1.aln -out_lib=sample_lib1.tc_lib -lib_only

Command 2:
$$: t_coffee -aln=sample_aln1.aln -out_lib=sample_lib1.tc_lib -lib_only -weight=1000
```

8.2.14 Q: I want to concatenate two libraries

A: You cannot concatenate these files on their own. You will have to use T-Coffee assuming you want to combine for instance `tc_lib1.tc_lib` and `tc_lib2.tc_lib`:

```
$$: t_coffee -lib=sample_lib1.tc_lib,sample_lib2.tc_lib -lib_only -out_lib=sample_
↪lib3.tc_lib
```

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

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

8.2.16 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:

```
Postscript
$$: t_coffee sample_seq1.fasta -output=score_ps

PDF only if you have ps2pdf installed
$*: t_coffee sample_seq1.fasta -output=score_pdf
```

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

A: See the next question.

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

```
$$: t_coffee sample_seq1.fasta -output=clustalw,html,score_ps,msf
```

Note: 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.

8.3 Alignment computation

8.3.1 Q: Is T-Coffee the best? Why not using MUSCLE, MAFFT, or ProbCons???

A: All these packages are good packages and they sometimes outperform T-Coffee. They also claim to outperform one another... If you have them installed locally, you can have T-Coffee to generate a consensus alignment:

```
$$: t_coffee sample_seq1.fasta -method muscle_msa,probcons_msa,mafft_msa,lalign_id_
↪pair,slow_pair
```

8.3.2 Q: Can T_Coffee align nucleic acids ???

A: Normally it can, but check in the log that the program recognises the right type. If this fails, you will need to manually set the type using **-type dna**

8.3.3 Q: I do not want to compute the alignment

A: use the **-convert** flag. This command will read the .aln file and turn it into an .msf alignment.

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

8.3.4 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**. You can either specify single (command 1) or multiple constraints using a TC_LIB_FORMAT_02 file (command 2). When giving 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. **T-Coffee Technical Documentation**, subsection **Creating your own T-Coffee libraries**).

```
Command 1: single constraint
$$: t_coffee -other_pg seq_reformat -in sample_aln3.aln -action +force_aln seq1 5␣
↪seq2 6
```

(continues on next page)

(continued from previous page)

```
Command 2: multiple constraints
$$: t_coffee -other_pg seq_reformat -in sample_aln3.aln -action +force_aln sample_
↪lib3.tc_lib02
```

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. The tc_lib02 format is as follow:

```
*TC_LIB_FORMAT_02
SeqX resY ResY_index  SeqZ ResZ ResZ_index
```

8.3.5 Q: I would like to use structural alignments

Refer to the **T-Coffee Main Documentation and/or T-Coffee Technical Documentation**.

8.3.6 Q: I want to build my own libraries

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

```
$$: t_coffee -aln=sample_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:

```
$$: t_coffee -seq=sample_seq1.fasta -lib=sample_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.

8.3.7 Q: I want to use my own tree

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

```
$$: t_coffee sample_seq1.fasta -usetree=sample_seq1_tree_nj.nwk
```

8.3.8 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. Notice that in the resulting alignments (command 1), all the gaps are of modulo3, except one small gap in the first line of sequence hmgl_trybr. This is a frameshift made on purpose. You can realign the same sequences while ignoring their coding potential and treating them like standard DNA (command 2).

```
Command 1:
$$: t_coffee three_cdna.fasta -method=cdna_fast_pair

Command 2:
$$: t_coffee three_cdna.fasta
```

Warning: This method has not yet been fully tested and is only provided 'as-is' with no warranty. Any feedback will be much appreciated.

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

See next question.

8.3.10 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. Pairwise methods (slow_pair, proba_pair, <method>_pair...) will only be applied to list of pairs of sequences, while multiple methods (clustalw_msa, mafft_msa, <method_msa...) will be applied to any dataset having more than two sequences.

```
$$: t_coffee sample_seq1.fasta -method=clustalw_pair,clustalw_msa -lib_list=sample_
↪list1.lib_list

Format of the list of libraries:
***************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****
```

8.3.11 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 overweighting of overrepresented sequences.

8.4 Using Structures and Profiles

8.4.1 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:

```
$$: t_coffee sample_seq1.fasta -profile=sample_aln2.aln -outfile chewbacca
```

8.4.2 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:

```
$$: t_coffee -profile=sample_aln1.aln,sample_aln2.aln -outfile han_solo
```

8.4.3 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:

```
$$: t_coffee -profile=sample_profile1.aln,sample_profile2.aln -mode=3dcoffee
```

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

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

```
Variables to set up:
##: setenv (or export) PDB_DIR='directory containing the pdb structures'
##: setenv (or export) 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. . . hummmm. . . strange.. Then again I was never struck by the efficiency of the Bush Jr administration.

8.4.5 Q: Can I use a local installation of PDB?

A: Yes, T-Coffee supports three types of installations:

- *Ad hoc* installation where all your structures are in a directory under the form pdbid.pdb, pdbid.id.Z or pdbid.pdb.gz. In that case, all you need to do is set the environement variables correctly:

```
Setting up variable
##: setenv (or export) PDB_DIR='directory containing the pdb structures'
##: setenv (or export) NO_REMOTE_PDB_DIR=1
```

- Full standard PDB installation using the all section of PDB. In that case, you must set the variables to:

```
Setting up variable
##: setenv (or export) PDB_DIR='<some absolute path>/data/structures/all/pdb/'
##: setenv (or export) NO_REMOTE_PDB_DIR=1
```

- Reduced standard PDB installation using the divided section of pdb:

```
Setting up the PDB:
##: setenv (or export) PDB_DIR='<some absolute path>/data/structures/divided/pdb/
##: setenv (or export) NO_REMOTE_PDB_DIR=1
```

If you need to do more clever things, you should know that all the PDB manipulation is made in T-Coffee by a perl script named **extract_from_pdb**. You can then edit the script to suit your needs; T-Coffee will use your edited version if it is in the current directory and issue a warning that it used a local version. If you make extensive modifications, I would appreciate you send me the corrected file so that I can incorporate it in the next distribution. By default, T-Coffee also requires two important PDB files declared using the two following variables. These variables do not need to be set if the considered files are in the cache directory (default behavior):

```
Found at: ftp://ftp.wwpdb.org/pub/pdb/derived_data/pdb_entry_type.txt
##: export PDB_ENTRY_TYPE_FILE=<location of the file pdb_entry_type.txt>
```

(continues on next page)

(continued from previous page)

```
Found at: http://www.rcsb.org/pdb/rest/getUnreleased
##: export PDB_UNREALEASED_FILE=<location of the file unrealeased.xml>
```

Warning: Since the file `unreleased.xml` is not part of the PDB distribution, T-Coffee will make an attempt to obtain it even when using the **NO_REMOTE_PDB_DIR=1 mode**. You must therefore make sure that the file `PDB_UNREALEASED_FILE` is pointing to is read and write.

8.5 Improving/Evaluating Your MSAs

8.5.1 Q: How can I edit my alignment manually?

A: We recommend to use Jalview, a free program for MSA editing that you can find here.

8.5.2 Q: Have I improved or not my alignment?

A: Using structural information is the only way to establish whether you have improved or not your alignment. The CORE index can also give you some information. Refers to the **T-Coffee Main Documentation**, section **Evaluating Your Alignment**.

8.5.3 Q: How good is my alignment?

A: Refers to the **T-Coffee Main Documentation**, section **Evaluating Your Alignment**. Or just look at the color index ;-)

8.5.4 Q: What is that color index?

A: T-Coffee can provide you with a measure of consistency among all the methods used. An html file is produced by default each time you run an alignment. This html file is a colored version of your MSA that you can visualize with any common browser. As alternatives, you can use **score_ps** (postscript), **score_pdf** (pdf file) or **score_ascii** (text file). If you want more information about the CORE index represented by this color index, have a look at this chapter.

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

A: Yes !! You may have an alignment produced from any source you like. If you have no library available, the library will be computed on the fly but this can take some time depending on your sample size.

```
With a library:
$$: t_coffee -infile=sample_aln1.aln -lib=sample_aln1.tc_lib -special_mode=evaluate

Without a library:
$$: t_coffee -infile=sample_aln1.aln -evaluate -method proba_pair
```

8.5.6 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.

```
$$: t_coffee -infile=sample_aln1_1.aln -aln=sample_aln1_2.aln -special_mode=evaluate
```

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

A: Increase the ktuple size (up to 4-5 for DNA) and up to 3 for proteins. This will speed up the program. It can be very useful, especially when aligning ESTs.

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

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

A: If there is no duplicated name in your sequence set, T-Coffee handles names similarly to Clustalw. If your dataset contains sequences with identical names, these will automatically be renamed by adding an index (integer) to duplicated names even if there are more than 2. Also be careful, if there are spaces in your names, whatever comes after the space is not read.

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

8.6 Release Notes

Warning: This log of modifications is not as thorough and accurate as it should be... but it's a beginning !

- 11.00+: extensive update of the documentation, examples; addition of the latest T-Coffee modes (PSI-Coffee, SARA-Coffee, Pro-Coffee, STRIKE, T-RMSD...); creation of an automated procedure for checking command lines from the documentation **doc2test.pl**.
- 9.86 New data structure for the primary library that results in highly improved running times for mcoffee and significantly decreased memory usage.
- 5.80 Novel assembly algorithm (linked_pair_wise) and the primary library is now made of probcons style pairwise alignments (proba_pair)
- 4.30 and upward: the FAQ has moved into a new tutorial document; **-in** can be replaced by the flags **-profile,-method,-aln,-seq,-pdb**.
- 4.02: **-mode=dna** is still available but not any more needed or supported. Use **-type=protein or dna** if you need to force things
- 3.28: corrected a bug that prevents short sequences from being correctly aligned
- 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

CHAPTER 9

T-Coffee release guide

9.1 Introduction

Before the realease of a new T-Coffee version, many command lines in the documentation are programmatically tested. The purpose of this document is to explain how to make a new release, which command lines are tested, how new command lines can be added to the documentation and how to perform a documentation check before a new release.

9.2 Making a new release

T-Coffee is automatically build by using the [Circle CI](https://circleci.com/gh/cbcrg/tcoffee) service. The procedure starts with an update of the version number followed by a git push that triggers the validation of the new release on circleCI and a posting via an Amazon instance hosting the T-Coffee web site. All these operations are detailed below. Depending on the release type the correspinding counter will be incremented by 1 <nmajor>.<stable>.<beta>

```
$#: cd t_coffee/src
$#: make release type=beta(default)|stable|major m='short description of the main␣
↪feature' test=core(default)|all|full|remote|docs|web|none
```

core : will test a throrough set of commands all : will run all the test commands packaged as dumps in tests full : will test all the commands packaged as dumps in the git repo - they are normally in docs and tests remote: will run all tze tests command dealing with remote access (BLAST and PDB) docs : will run all the dumps compiled from the rst files

9.2.1 Beta Release

For a beta, the following needs to go through:

```
$#: make release type=beta test=core|none
```

9.2.2 Stable Release

For a Stable, the following needs to go through:

```
$#: make release type=beta test=core
$#: make release type=stable teste=none
```

9.2.3 Major Release

For a Major, the following needs to go through. Note that a major implies some important milestone such as a publication. For a start, you should make sure the documentation compiles well. If needed you should recompile all the available tests

```
$#: doc2test.pl -play docs/doc2test.rst -data docs/.data/ -dump docs/.dumps/ -update
$#: doc2test.pl -play tests/all2test.rst -data tests/.data/ -dump tests/.dumps/ -
↪update
```

Tip: doc2test contains the list of ALL the rst files to be tested. Note that in an RST files, the only files CL that get tested are the ones starting wit **$$**

Tip: The files required tu run the command lines are available in docs/.data and test/.data

Make sure ALL the tests pass well on your local machine before running them on CircleCI

```
$#: doc2test.pl -replay  .
```

If they do, you should the run the following

```
$#: make release type=beta test=full
```

If this beta is succesful, then you can pack the major

```
$#: make release type=major teste=none
```

9.3 Release Building Procedure

This section explains how the release procedure works. Roughly speaking, the release is initiated on a local machine using the makefile (t_coffee/src/makefile). It is this procedure that increments the release version number (lib/version/version_number.version). This procedure makes a dummy distribution on the local machine and eventually pushes all the files in github. Thanks to a CircleCI hook, the push then initiates the building of a virtual machine in CircleCI where the final building procedure is carried out amd followed by a test (as specifiled in makefile test=<test type>. If the test is succesful, the procedure finishes with a publishing of the distribution.

9.3.1 Triggering a release

- go into t_coffee/src
- type make release=beta|stable|major m='commit comments' – including [ci skip] if needed

This will trigger - An update of the version number in lib/version/version_number.version: Version_<major>.<stable>.<build>.<github Branch Number> - The file .circleci/config.yml will be programmatically edidted by read_program_version.pl to state the kind of release to be done by CircleCI - The command 'make distribution' will run. Among othe things it will

- Check that the distribution procedure is functional - it is a similar procedure that will be used by CircleCI
- Compile the documentation with sphinx and update docs/.html – this requires sphinx to be installed on the local machine

9.3.2 CircleCI building

The tcoffee git repository is registered to CircleCI via the following hook: https://github.com/cbcrg/tcoffee/settings/hooks

The Circle build is controlled by the [.circleci/circle.yml](circle.yml) file and processes as follows:

1. The T-Coffee build/build.sh script is executed in the [cbcrg/tcoffee-build-box:1.2](docker/Dockerfile.buildbox) container
2. The T-Coffee binaries produced by the build process are created in the folder *~/publish/sandbox/build*. This folder is moved under path *build/tcoffee*
3. A new Docker image named *xcoffee* is created copying the content of the folder *build/tcoffee*.The image is built by using this docker/Dockerfile.
4. The new *xcoffee* build is tested by running the docker/run-tests.shtests suite.
5. The summary of the tests is available on https://circleci.com/gh/cbcrg/tcoffee/tree/master
6. If all tests are passed the *xcoffee* image is pushed to the [Docker Hub](https://hub.docker.com/r/cbcrg/tcoffee/tags/) with the names *cbcrg/tcoffee:latest* and *cbcrg/tcoffee:<version.commit-id>*
7. The environment variable *RELEASE=0|1* is used to mark the build as beta or stable (use the [build/make_release.sh] to trigger a new release build).

9.3.3 Publication

Once the build is complete and all tests are passed the distribution is pushed onto the web along with the associated documentation.

The utility doc2test.pl makes it possible to run a command line on data located in a folder and to produce a dump file that will then be a stand alone tests, that can be transfered and replayed. The dump file is a container in which the input, the stdoin, stdout, environement and output of the run are recorded.

Tests are systematically carried out on all the command lines that start with the symbol $$. For instance, the following CL will has been tested.

```
$#: t_coffee
```

Other command lines starting with different symbols are not checked. Two types of command lines identifiers are used:

```
$#: t_coffee
```

For a command that could be tested but will not be, either for the sake of time or because it is currently unstable. When a new release needs to be urgently made available because of a critical fix, it is advisable to comment out this way non critical command lines failing the test.

```
$#: t_coffee
```

These commands are never tested, either because they contain system dependant information or non programmatic information.

Whenvever adding a new command, input files must be added to the repository directory ./examples/. New commands can be also be built using the existing files, or they can depend on files newly added to the repository.

9.3.4 Using doc2test.pl

The utility doc2test is meant to do the following actions:

The play action will compile dump files.

```
#$: ./lib/perl/lib/perl4makefile/doc2test.pl -play <*.rst file or *.tests␣
↪file or list of cmd in txt> -data <folder containing the data> -dumps
↪<target directory for dumps>
```

The Check action will recursively go through any directory and report the exit status of each dump

```
#$: ./lib/perl/lib/perl4makefile/doc2test.pl -check <dump directory - recursive>
```

Note that while compiling Command Lines, you can use check to clean your directory and remove any failed dump.

```
#$: ./lib/perl/lib/perl4makefile/doc2test.pl -check <dump directory - recursive> -
↪clean FAILED or -clean2 <string in dump to be deleted>
```

You should also use check to make a fresh build and remove all previous dumps:

```
#$: ./lib/perl/lib/perl4makefile/doc2test.pl -check <dump directory - recursive> -
↪clean ALL
```

And you can the re-run the play that will only regenerate missing dumps:

```
#$: ./lib/perl/lib/perl4makefile/doc2test.pl -play <*.rst file or *.tests file or␣
↪list of cmd in txt> -data <folder containing the data> -dumps <target directory for␣
↪dumps>
```

The replay action will re-run each dump and report the status. It will report new warnings.

```
#$: ./lib/perl/lib/perl4makefile/doc2test.pl -replay <dump directory - recursive>
```

The unplay option maked it possible to output the inoput files of every dump

```
#$: ./lib/perl/lib/perl4makefile/doc2test.pl -replay <dump directory - recursive> -
↪outdir <file in which data should be dumped>
```

9.3.5 Compiling tests from the rst documentation

The rst documentation contain lines tagged with two $$ signs. These lines will be selected for validation and used to build dump files. In order to run a fresh collection of dumps for the full documentation, run the foillowing commands:

The simplest way to compile all possible tests from the the documentation is to run the following command from the top of the git repo.

```
#$: ./lib/perl/lib/perl4makefile/doc2test.pl -check docs/.dumps -clean ALL
#$: ./lib/perl/lib/perl4makefile/doc2test.pl -play docs/doc2test.rst -data␣
↪docs/.data -dump docs/.dumps
```

The output can then be checked

```
#$: ./lib/perl/lib/perl4makefile/doc2test.pl -check docs/.dumps
```

The failing jobs can be removed and re-run once the issue has been identified (usualy a missing file):

```
#$: ./lib/perl/lib/perl4makefile/doc2test.pl -check docs/.dumps -clean FAILED
```

Note that in order to figure out an issue you can use the replay debug option

```
#$: ./lib/perl/lib/perl4makefile/doc2test.pl -replay <dump file> -debug
```

You can the re-run all the dumps. This is how the tests will be carried out by CircleCI

```
#$: ./lib/perl/lib/perl4makefile/doc2test.pl -replay docs/.dumps
```

9.3.6 Compiling tests commands

tests are special types of pre-compiled commands used to validate specific components of T-Coffee. You can compile and use thsis test in the same way as rst files.

9.3.7 Creating a new command to be tested

In order to generate a new test command, all you need to do is to run your command while setting an environement variable. This will generate a dump file that can be used torerun the same call. Dump files are very easy to produce yout simply need to run

```
##:rm -f <your dump file>;export DUMP_4_TCOFFEE=<your dump file>;t_coffee -in seq
OR
##:rm -f <your dump file>;export DUMP_4_TCOFFEE=<your dump file>;t_coffee -other_pg␣
↪seq_reformat -in xxxx/xx/s.pep -output fasta_seq > yyy
```

This command will generate a self contained dumpfile. This dumpfile will contain all the information needed ton reprodduce the runn (i.e file name, path, content and T-Coffee parameters). In order to reproduce the call you need to have T-Coffee installed and run:

```
##: ./lib/perl/lib/perl4makefile/doc2test.pl -replay yourdumpfile
```

Will allow you to check if the command runs and produces similar files. By default the test is "PASSED" if no error is thrown by the call. This can be refined as follows:

##: ./lib/perl/lib/perl4makefile/doc2test.pl -replay yourdumpfile -strict

Will report failure whenever an output file is missing or whenever an error is reported.

##: ./lib/perl/lib/perl4makefile/doc2test.pl -replay yourdumpfile -very_strict

Will report failure whenever there is a warning OR whenever an output file differs (Note that VERSION and CPU are excluded from the comparison as a consequence different versions will not result in different output files). It is possible to visualize the replayed dump for debugging purposes.

##: ./lib/perl/lib/perl4makefile/doc2test.pl -replay yourdumpfile -keepreplayed

CHAPTER 10

Release Notes

Warning: This log of recent modifications is not as thorough and accurate as it should be.

-14.00 new faster regressive algorithm

-9.86 New data structure for the primary library that results in highly improved running times for mcoffee and significantly decreased memory usage.

-5.80 Novel assembly algorithm (linked_pair_wise) and the primary library is now made of probcons style pairwise alignments (proba_pair)

-4.30 and upward: the FAQ has moved into a new tutorial document

-4.30 and upward: -in has will be deprecated and replaced by the flags: -profile,-method,-aln,-seq,-pdb

-4.02: -mode=dna is still available but not any more needed or supported. Use type=protein or dna if you need to force things

-3.28: corrected a bug that prevents short sequences from being correctly aligned

-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

CHAPTER 11

Citations & References

Note: 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 deliver donuts to our offices from time to time (Not that they ever did it, but it would be nice anyway). Cite the approriate web server if you use one, otherwise, cite the original paper from 2000 (By the way...NOOO, it was never named 'T-Coffee 2000').

11.1 Comparative Genomics & Biology

Erb, I., Notredame, C. **How should we measure proportionality on relative gene expression data?** Theory Biosci., 135(1-2):21-36 (2016), PMID26762323

Catuara-Solarz, S., Espinosa-Carrasco, J., Erb, I., Langohr, K., Gonzalez, J.R., Notredame, C., Dierssen, M. **Combined treatment with environmental enrichment and (-)-epigallocatchin-3-gallate ameliorates learning deficits and hippocampal alterations in a mouse model of Down syndrome**. eNeuro., 3(5)(2015) pii:ENEURO.0103, PMID:27844057

Di Tommaso, P., Palumbo, E., Chatzou, M., Prieto, P.,Heuer, M.L., Notredame, C. **The impact of Docker containers on ther performance of genomic pipelines**. PeerJ., 3:e1273 (2015), PMID:26421241

Catuara-Solarz, S., Espinosa-Carrasco, J., Erb, I., Langohr, K., Notredame, C., Gonzalez, J.R., Dierssen, M. **Principal component analysis of the effects of environmental enrichment and (-)-epigallocatchin-3-gallate on age-associated learning deficits in a mouse model of Down syndrome**. Front Behav. Neurosci., 9:330 (2015), PMID:26696850

Earl, D., Nguyen, N., Hickey, G., Harris, R.S., Fitzgerald, S., Beal, K. Seledtsov, I., Raney, B.J., Clawson, H., Kim, J., Kemena, C., Chang, J.-M., Erb, I., Poliakov, A., Hou, M., Herrero, J., Kent, W.J., Solovyev, V., Darling, A.E., Ma, J., Notredame, C., Brudno, M., Dubchak, I., Haussler, D., Paten, B. **Alignathon: a competitive assessment of whole-geome alignment methods**. Genome Res., 24(12):2077-2089 (2014), PMID:25273068

The mouse ENCODE Consortium. **A comparative encyclopedia of DNA elements in the mouse genome**. Nature, 515(7527):355-364 (2014), PMID:25409824

Bussotti, G., Notredame, C., Enright, A.J. **Detecting and comparing non-coding RNAs in the high-throughput era**. Int. J. Mol. Sci., 14(8):15423-15458 (2013), PMID:23887659

Chang, J.-M., Taly, J.-F., Erb, I., Sung, T.Y., Hsu, W.L. Tang, C.Y., Notredame, C., Su, E.C. **Efficient and interpretable prediction of protein functional classes by correspondence analysis and compact set relations**. PLoS One, 8(10):e75542 (2013), PMID:24146760

Magis, C., van der Sloot, A.M., Serrano, L., Notredame, C. **An improved understanding of TNFL/TNFR interactions using structure-based classifications**. Trends Biochem. Sci., 37(9):353-363 (2012), PMID:22789664

Derrien, T., Johnson, R., Bussotti, G., Tanzer, A., Djebali, S., Tilgner, H., Guernec, G., Martin, D., Merkel, A., Knowles, D.G., Lagarde, J., Veeravalli, L., Ruan, X., Ruan, Y., Lassmann, T., Carninci, P., Brown, J.B., Lipovich, L., Gonzalez, J.M., Davis, C.A., Shiekhattar, R., Gingeras, T.R., Hubbard, T.J., Notredame, C., Harrow, J., Guigó, R. **The GENCODE v7 catalog of human long noncoding RNAs: analysis of their gene structure, evolution and expression**. Genome Res., 22(9):1775-1789 (2012), PMID:22955988

Breen, M.S., Kemena, C., Vlasov, P.K., Notredame, C., Kondrashov, F.A. **Epistasis as the primary factor in molecular evolution**. Nature, 490(7421), 535-538 (2012), PMID:23064225

Bussotti, G., Raineri, E., Erb, I., Zytnicki, M., Wilm, A., Beaudoing, E., Bucher, P., Notredame, C. **BlastR-fast and accurate database searches for non-coding RNAs**. Nucleic Acids Res., 39(16):6886-6895 (2011), PMID:21624887

Lin, H.N., Notredame, C., Chang, J.-M., Sung, T.Y., Hsu, W.L. **Improving the alignment quality of consistency based aligners with an evaluation function using synonymous protein words**. PLoS One, 6(12):e27872, (2011), PMID:22163274

Ørom, U.A., Derrien, T., Beringer, M., Gumireddy, K., Gardini, A., Bussotti, G., Lai, F., Zytnicki, M., Notredame, C., Huang, Q., Guigo, R., Shiekhattar, R. **Long noncoding RNAs with enhancer-like function in human cells**. Cell, 143(1):46-58 (2010), PMID:20887892

Coll, O., Villalba, A., Bussotti, G., Notredame, C., Gebauer, F. **A novel, noncanonical mechanism of cytoplasmic polyadenylation operates in Drosophila embryogenesis**. Genes Dev., 24(2):129-134 (2010), PMID:20080951

Rausch, T., Emde, A.K., Weese, D., Döring, A., Notredame, C., Reinert, K. **Segment-based multiple sequence alignment**. Bioinformatics, 24(16):i187-i192 (2008), PMID:18689823

Dietmann, S:, Park, J., Notredame, C., Heger, A., Lappe, M., Holm, L. **A fully automatic evolutionary classification of protein folds: Dali Domain Dictionary version 3**. Nucleic Acids Res., 29(1):55-57 (2001), PMID:11125048

11.2 T-Coffee Alignment Methods

Kemena, C., Bussotti, G., Capriotti, E., Marti-Renom, M.A., Noredame, C. **Using tertiary structure for computation of highly accurate multiple RNA alignments with the SARA-Coffee package**. Bioinformatics, 29(9):1112-1119 (2013), PMID:234490094

Erb, I., González-Vallinas, J.R., Bussotti, G., Blanco, E., Eyras, E., Notredame, C. **Use of ChIP-Seq data for the design of a multiple promoter-alignment method**. Nucleic Acids Res., 40(7):e52 (2012), PMID:22230796

Chang, J.-M., Di Tommaso, P., Taly, J.-F., Notredame, C. **Accurate multiple sequence alignment of transmembrane proteins with PSI-Coffee**. BMC Bioinformatics, 13 Suppl 4:S1 (2012). PMID:22536955

Di Tommaso, P., Orobitg, M., Guirado, F., Cores, F., Espinosa, T., Notredame, C. **Cloud-Coffee: implementation of a parallel consistency-based multiple alignment algorithm in the T-Coffee package and its benchmarking on the Amazon Elastic-Cloud**. Bioinformatics, 26(15):1903-1904 (2010). PMID:20605929

Wilm, A., Higgins, D.G., Notredame, C. **R-Coffee: a method for multiple alignment of non-coding RNA**. Nucleic Acids Res., 36(9):e52 (2008), PMID:18420654

Armougom, F., Moretti, S., Poirot, O., Audic, S., Dumas, P., Schaeli, B., Keduas, V., Notredame. C. **Expresso: automatic incorporation of structural information in multiple sequence alignments using 3D-Coffee**. Nucleic Acids Res., 34:W604-W608 (2006), PMID:16845081

Wallace, I.M., O'Sullivan, O., Higgins, D.G., Notredame, C. **M-Coffee: combining multiple sequence alignment methods with T-Coffee**. Nucleic Acids Res., 34(6):1692-1699 (2006), PMID:16556910

O'Sullivan, O., Suhre, K., Abergel, C., Higgins, D.G., Notredame, C. **3DCoffee: combining protein sequences and structures within multiple sequence alignments**. J. Mol. Biol., 340(2):385-395 (2004). PMID: 15201059

O'Sullivan, O., Zehnder, M., Higgins, D.G., Bucher, P., Grosdidier, A., Notredame, C. **APDB: a novel measure for benchmarking sequence alignment methods without reference alignments**. Bioinformatics, 19 Suppl 1:i215-i221 (2003). PMID:12855461

Notredame, C. **Mocca: semi-automatic method for domain hunting**. Bioinformatics, 17(4):373-374 (2001). PMID: 11301309

Notredame, C., Higgins, D.G., Heringa, J. **T-Coffee: A novel method for fast and accurate multiple sequence alignment**. J. Mol. Biol., 302(1):205-217 (2000). PMID: 10964570

Notredame, C., Holm, L., Higgins, D.G. **COFFEE: an objective function for multiple sequence alignments**. Bioinformatics, 14(5):407-422 (1998). PMID: 968205

11.3 T-Coffee Web Servers

Floden, E.W., Tommaso, P.D., Chatzou, M., Magis, C., Notredame, C., Chang, J.-M. **PSI/TM-Coffee: a web server for fasta accurate multiple sequence alignments of regular and transmembrane proteins using homology extension on reduced databases**. Nucleic Acids Res., 44(W1):W339-W343 (2016). PMID:27106060

Chang, J.-M.*, Tommaso P.*, Lefort, V., Gascuel, O., Notredame, C. **TCS: a web server for multiple sequence alignment evaluation and phylogenetic reconstruction**. Nucleic Acids Research. 43(W1):W3-6 (2015). PMID: (*These two authors contribute equally)

Di Tommaso, P., Bussotti, G., Kemena, C., Capriotti, E., Chatzou, M., Prieto, P., Notredame, C. **SARA-Coffee web server, a tool for computation of RNA sequence and structure multiple alignments**. Nucleic Acids Res., 42(Web Server issue):W356-W360 (2014). PMID:24972831

Magis, C., Di Tommaso, P., Notredame, C. **T-RMSD: a web server for automated fine-grained protein structural classifcation**. Nucleic Acids Res., 41(Web Server issue):W358-W362 (2013). PMID:23716642

Rius, J., Cores, F., Solsona, F., van Hemert, J.I., Koetsier, J., Notredame, C. **A user-friendly web portal for T-Coffee on supercomputers**. BMC Bioinformatics, 12:150 (2011). PMID:21569428

Di Tommaso, P., Moretti, S., Xenarios, I., Orobitg, M., Montanyola, A., Chang, J.-M., Taly, J.-F., Notredame, C. **T-Coffee: a web server for the multiple sequence alignment of protein and RNA sequences using structural information and homology extension**. Nucleic Acids Res., 39(Web Server issue):W13-W17 (2011). PMID:21558174

Moretti, S., Wilm, A., Higgins, D.G., Xenarios, I., Notredame, C. **R-Coffee: a web server for accurately aligning noncoding RNA sequences**., Nucleic Acids Res., 36(Web Server issue):W10-W13 (2008). PMID:18483080

Moretti, S., Armougom, F., Wallace, I.M., Higgins, D.G., Jongeneel, C.V., Notredame, C. **The M-Coffee web server: a meta-method for computing multiple sequence alignments by combining alternative alignment methods**. Nucleic Acids Res., 35(Web Server issue):W645-W648 (2007). PMID:17526519

Moretti, S., Reinier, F., Poirot, O., Armougom, F., Audic, S., Keduas, V., Notredame, C. **PROTOGENE: turning amino acid alignments into bona fide CDS nucleotide alignments**. Nucleic Acids Res., 34(Web server issue):W600-W603 (2006), PMID:16845080

Armougom, F., Moretti, S., Keduas, V., Notredame, C. **The iRMSD: a local measure of sequence alignment accuracy using structural information**. Bioinformatics, 22(14):e35-e39 (2006). PMID:16873492

Armougom, F., Poirot, O., Moretti, S., Higgins, D.G., Bucher, P., Keduas, V., Notredame, C. **APDB: a web server to evaluate the accuracy of the sequence alignments using structural information**. Bioinformatics, 22(19):2439-2440 (2006). PMID:17032685

Claude, J.B., Suhre, K., Notredame, C., Claverie, J.M., Abergel C. **CaspR: a web server for automated molecular replacement using homology modelling**. Nucleic Acids Res., 32(Web Server issue):W606-9 (2004). PMID:15215460

Poirot, O., Suhre, K., Abergel, C., O'Toole, E., Notredame, C. **3DCoffee@igs: a web server for combining sequences and structures into a multiple sequence alignment**. Nucleic Acids Res., 32(Web Server issue):W37-40 (2004). PMID: 15215345

Poirot, O., O'Toole, E., Notredame, C. **Tcoffee@igs: a web server for computing, evaluating and combining multiple sequence alignments**. Nucleic Acids Res., Jul 1;31(13):3503-6 (2003). PMID: 12824354

11.4 T-Coffee Tools For MSA Evaluation

Chang, J.-M., Tommaso, P., Notredame, C. **TCS: a new Multiple Sequence Alignment reliability measure to estimate alignment accuracy and improve phylogenetic tree reconstruction**. Molecular Biology and Evolution, 31(6):1625-37 (2014). PMID:24694831

Kemena, C., Taly, J.-F., Kleinjung, J., Notredame, C. **STRIKE: evaluation of protein MSAs using a single 3D structure**. Bioinformatics, 27(24):3385-3391 (2011). PMID:220339207

Magis, C., Stricher, F., van der Sloot, A.M., Serrano, L., Notredame, C. **T-RMSD: a fine-grained, structure based classification method and its application to the functional characterization of TNF receptors**. J. Mol. Biol., 400(3):605-617 (2010), PMID:20471393

Armougom, F., Moretti, S., Keduas, V., Notredame, C. **The iRMSD: a local measure of sequence alignment accuracy using structural information**. Bioinformatics, 22(14):e35-e39 (2006), PMID:16873492

11.5 Reviews, Briefings & Books

Chatzou, M., Magis, C., Chang, J.-M., Kemena, C., Bussotti, G., Erb, I., Notredame, C. **Multiple sequence alignment modeling: methods and applications**. Brief Bioinform., pii: bbv099 (2015). PMID:26615024

Magis, C., Taly, J.-F., Bussotti, G., Chang, J.M., Di Tommaso, P., Erb, I., Espinosa-Carrasco, J., Notredame, C. **T-Coffee: tree-based consistency objective function for alignment evaluation**. Methods Mol. Biol., 1079:117-129 (2014). PMID:24170398

Taly, J.-F.*, Magis, C.*, Bussotti, G., Chang, J.-M., Di Tommaso, P., Erb, I., Espinosa-Carrasco, J., Kemena, C., Notredame, C. **Using the T-Coffee package to build multiple sequence alignments of protein, RNA, DNA sequences and 3D structures**. Nature Protocols, 1669-1682 (2011), PMID:21979275 (*These two authors contribute equally)

Notredame, C. **Computing multiple sequence/structure alignments with the T-Coffee package**. Curr. Protoc. Bioinformatics, Chapter 3, Unit 3.8.1-25 (2010). PMID:20205190

Kemena, C., Notredame, C. **Upcoming challenges for multiple sequence alignment methods in the high-throughput era**. Bioinformatics, 25(19):2455-2465 (2009). PMID:19648142

Notredame, C. **Recent evolutions of multiple sequence alignment algorithms**. PLoS Comput. Biol., 3(8):e123 (2007). PMID:17784778

Notredame, C., Suhre, K. **Computing multiple sequence/structure alignments with the T-Coffee package**. Curr. Protoc. Bioinformatics, Chapter 3:Unit 3.8 (2004), PMID:18428722

Notredame, C., Abergel, C. **Using multiple alignment methods to assess the quality of genomic data analysis**. In: Andrade MA, editor. Bioinformatics and genomes: current perspectives. Wymondham (UK): Horizon Scientific Press,30–50 (2003).

Notredame, C. **Recent progress in multiple sequence alignment: a survey**. Pharmacogenomics, 3(1):131-144 (2002). PMID:11966409

CHAPTER 12

Contacts, Adresses & Contributions

12.1 Contact us

If you have any problem, question, suggestion or else, contact Cedric Notredame by mail at cedric.notredame@crg.eu or at the T-Coffee group tcoffee@googlegroups.com.

12.2 Adresses

The T-Coffee group is currently at the Centre for Genomic Regulation (CRG), programme of Bioinformatics and Genomics (BiG), in Barcelona (PRBB, 88 calle del Doctor Aiguader, 08003, Barcelona). We are always very eager to get some user feedback, so please do not hesitate to drop us a line at cedric.notredame@crg.eu. The latest updates of T-Coffee are always available on our group webpage where you will also find links to all the online T-Coffee servers.

12.3 Contributions

12.3.1 Main Contributions

T-coffee is developed, maintained, monitored, used and debugged by a dedicated team that include or have included:

Cedric Notredame, Paolo di Tommaso, Jean-François Taly, Cedrik Magis, Fabrice Armougom, Des Higgins, Sebastien Moretti, Orla O'Sullivan, Eamon O'Toole, Olivier Poirot, Karsten Suhre, Iain Wallace, Andreas Wilm, Vladimir Keduas.

12.3.2 Other Contributions

We do not mean to steal code, but we will always try to re-use pre-existing code whenever that code exists, free of copyright, just like we expect people to do with our code. However, whenever this happens, we make a point at

properly citing the source of the original contribution. If ever you recognize a piece of your code improperly cited, please drop us a note and we will be happy to correct that.

12.3.2.1 Softwares

In the mean time, here are some important pieces of code from other packages that have been incorporated within the T-Coffee package. These include:

- TMalign package from Zhang, Jeffrey and Skolnik (NAR, 2005, 33:2303).
- 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
- David Mathog (Caltech) provided many fixes and useful feedback for improving the code and making the whole soft behaving more rationally.

An enormous thanks to these people who believe in free open source code.

12.3.2.2 Bug reports and feedback

- Prof David Jones (UCL) reported and corrected the PDB1K bug (now t_coffee/sap can align PDB sequences longer than 1000 AA).
- Johan Leckner reported several bugs related to the treatment of PDB structures, insuring a consistent behavior between version 1.37 and current ones.
- Enrico Bonatesta reported several mistakes in the T-Coffee tutorial and provided feedbacks about its readibility.
- . . . and to everyone else who send us their feedbacks and suggestions helping us to improve T-Coffee.

CHAPTER 13

License & 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 are very little restrictions 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 for the cut and paste of a few functions, although we are always happy to receive pieces of improved code.

T-Coffee can be incorporated in most pipelines: Plug-in/Plug-out...

Our philosophy is to insure that as many methods as possible can be used as plug-ins within T-Coffee. Likewise, we will give as much support as possible to anyone wishing to turn T-Coffee into a plug-in for another method. For more details on how to do this, see the plug-in and the plug-out sections of the Tutorial Manual.

Again, you do not need our permission to either use T-Coffee (or your method as a plug-in/out) but if you let us know, we will insure the stability of T-Coffee within your system through future releases.

The current license only allows for the incorporation of T-Coffee in non-commercial pipelines (i.e. where you do not sell the pipeline, or access to it). If your pipeline is commercial, please get in touch with us.

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.

```
$$: t_coffee -update
```