CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
CENTRO DE REGULACCIO GENOMICA, BARCELONA

Cédric Notredame

www.tcoffee.org

T-Coffee: Cheat Sheet Tutorial and FAQ

T-Coffee Tutorial (Version 6.18, August 2008) T-Coffee, PSI-Coffee 3D-Coffee/Expresso M-Coffee R-Coffee APDB and iRMSD

Cheat Sheet: T-Coffee 6
Proteins 6
DNA 6
RNA 6
Memory Problems 6

Before You Start.................................... 8
Foreword 8
Pre-Requisite 8
Getting the Example Files of the Tutorial 9

What Is T-COFFEE ?....................................10
What is T-Coffee? 10
What does it do?..10
What can it align?..10
How can I use it? 10
Is There an Online Server..11
Is T-Coffee different from ClustalW?..11
Is T-Coffee very accurate?..11
What T-Coffee Can and Cannot do for you … 12
(NOT) Fetching Sequences 12
Aligning Sequences 12
Combining Alignments..12
Evaluating Alignments..12
Combining Sequences and Structures..12
Identifying Occurrences of a Motif: Mocca..13
How Does T-Coffee works..13

Preparing Your Data: Reformatting and Trimming With seq_reformat....................................15
Seq_reformat..15
Accessing the T-Coffee Reformatting Utility 15
An overview of seq_reformat 16
Reformatting your data..16
Changing MSA formats..16
Dealing with Non-automatically recognized formats..16
Automated Sequence Edition..16
Removing the gaps from an alignment..16
Changing the case of your sequences 16
Changing the case of specific residues 17
Changing the case depending on the score..17
Protecting Important Sequence Names 17
Colouring/Editing Residues in an Alignment..18
Coloring specific types of residues 18
Coloring a specific residue of a specific sequence 18
Coloring according to the conservation 18
Colouring/Editing residues in an Alignment Using a Cache 19
Overview 19
Preparing a Sequence or Alignment Cache..19
Preparing a Library Cache..20
Coloring an Alignment using a cache..21
Changing the default colors 21
Evaluating an alignment and producing a cache..22
Evaluating an alignment with T-Coffee 22

Evaluating the level of conservation with a substitution matrix *22*
Selective Reformatting **23**
Removing gapped columns *23*
Selectively turn some residues to lower case *23*
Selectively modifying residues *24*
Keeping only the best portion of an alignment *24*
Extracting Portions of Dataset **25**
Extracting The High Scoring Blocks *25*
Extracting Sequences According to a Pattern *26*
Extracting Sequences by Names *26*
Removing Sequences by Names *27*
Extracting Blocks Within Alignment *27*
Concatenating Alignments *28*
Analyzing your Multiple Sequence Alignment **28**
Estimating the diversity in your alignment *28*
Reducing and improving your dataset **28**
Extracting the N most informative sequences *29*
Extracting all the sequences less than X% identical *29*
Speeding up the process *29*
Forcing Specific Sequences to be kept *30*
Identifying and Removing Outlayers *31*
Chaining Important Sequences *31*
Manipulating DNA sequences **31**
Translating DNA sequences into Proteins *31*
Back-Translation With the Bona-Fide DNA sequences *32*
Finding the Bona-Fide Sequences for the Back-Translation *32*
Guessing Your Back Translation *32*
Fetching a Structure **32**
Fetching a PDB structure *32*
Fetching The Sequence of a PDB structure *33*
Adapting extract_from_pdb to your own environment *33*

Manipulating RNA sequences with seq_reformat 34

Producing a Stockholm output: adding predicted secondary structures **34**
Producing a consensus structure *34*
Adding a consensus structure to an alignment *34*
Analyzing an alifold secondary structure prediction **35**
Analyzing matching columns *35*
Visualizing compensatory mutations *36*
Handling gapped columns *36*
Comparing alternative folds **36**

Manipulating Phylogenetic Trees with seq_reformat 37

Producing phylogenetic trees **37**
Comparing two phylogenetic trees **38**
Scanning Phylogenetic Trees **38**
Pruning Phylogenetic Trees **39**

Building Multiple Sequence Alignments 40

How to generate The Alignment You Need? **40**
What is a Good Alignment? *40*
The Main Methods and their Scope *41*
Choosing The Right Package *42*
Computing Multiple Sequence Alignments With T-Coffee **43**
Computing Very accurate (but slow) alignments with PSI-Coffee *43*
A Simple Multiple Sequence Alignment *43*

Controlling the Output Format......43
Computing a Phylogenetic tree......43
Using Several Datasets......44
How Good is Your Alignment......44
Doing it over the WWW......44
Aligning Many Sequences......45
Aligning Very Large Datasets with Muscle......45
Aligning Very Large Alignments with Mafft......45
Aligning Very Large Alignments with T-Coffee......45
Shrinking Large Alignments With T-Coffee......45
Modifying the default parameters of T-Coffee......45
Changing the Substitution Matrix......46
Comparing Two Alternative Alignments......46
Changing Gap Penalties......48
Can You Guess The Optimal Parameters?......49
Using Many Methods at once......49
Using All the Methods at the Same Time: M-Coffee......49
Using Selected Methods to Compute your MSA......50
Combining pre-Computed Alignments......51
Aligning Profiles......51
Using Profiles as templates......51
Aligning One sequence to a Profile......51
Aligning Many Sequences to a Profile......52
Aligning Other Types of Sequences......52
Splicing variants......52
Aligning DNA sequences......53
Aligning RNA sequences......53
Noisy Coding DNA Sequences.........53

Using Secondary Structure Predictions:......55
Single Sequence prediction......55
Multiple Sequence Predictions......55
Incorporation of the prediction in the alignment......56
Using other secondary structure predictions......56
Output of the prediction......57

Combining Sequences and 3D-Structures......58
If you are in a Hurry: Expresso......58
What is Expresso?......58
Using Expresso......59
Aligning Sequences and Structures......59
Mixing Sequences and Structures......59
Using Sequences only......60
Aligning Profile using Structural Information......60

How Good Is Your Alignment ?......61

Evaluating Alignments with The CORE index......61
Computing the Local CORE Index......61
Computing the CORE index of any alignment......61
Filtering Bad Residues......61
Filtering Gap Columns......62

Evaluating an Alignment Using Structural Information: APDB and iRMSD......63
What is the iRMSD?......63
How to Efficiently Use Structural Information......64
Evaluating an Alignment With the iRMSD Package......64

Evaluating Alternative Alignments *64*
Identifying the most distantly related sequences in your dataset *65*

Evaluating an Alignment according to your own Criterion ***65***
Establishing Your Own Criterion *65*

Integrating External Methods In T-Coffee 67

What Are The Methods Already Integrated in T-Coffee **67**
List of INTERNAL Methods *67*
Plug-In: Using Methods Integrated in T-Coffee *68*

Modifying the parameters of Internal and External Methods ***70***
Internal Methods *70*
External Methods *70*
Integrating External Methods **71**
Direct access to external methods *71*
Customizing an external method (with parameters) for T-Coffee *71*
Managing a collection of method files *72*
Advanced Method Integration **72**
The Mother of All method files… *74*
Weighting your Method *75*
Plug-Out: Using T-Coffee as a Plug-In **76**
Creating Your Own T-Coffee Libraries **76**
Using Pre-Computed Alignments *76*
Customizing the Weighting Scheme *76*
Generating Your Own Libraries *77*

Frequently Asked Questions 78

Abnormal Terminations and Wrong Results **78**
Q: The program keeps crashing when I give my sequences *78*
Q: The default alignment is not good enough *78*
Q: The alignment contains obvious mistakes *79*
Q: The program is crashing *79*
Q: I am running out of memory *79*
Input/Output Control **79**
Q: How many Sequences can t_coffee handle *79*
Q: Can I prevent the Output of all the warnings? *79*
Q: How many ways to pass parameters to t_coffee? *79*
Q: How can I change the default output format? *80*
Q: My sequences are slightly different between all the alignments *80*
Q: Is it possible to pipe stuff OUT of t_coffee? *80*
Q: Is it possible to pipe stuff INTO t_coffee? *80*
Q: Can I read my parameters from a file? *80*
Q: I want to decide myself on the name of the output files!!! *81*
Q: I want to use the sequences in an alignment file *81*
Q: I only want to produce a library *81*
Q: I want to turn an alignment into a library *81*
Q: I want to concatenate two libraries *81*
Q: What happens to the gaps when an alignment is fed to T-Coffee *82*
Q: I cannot print the html graphic display!!! *82*
Q: I want to output an html file and a regular file *82*
Q: I would like to output more than one alignment format at the same time *82*
Alignment Computation **83**
Q: Is T-Coffee the best? Why Not Using Muscle, or Mafft, or ProbCons??? *83*
Q: Can t_coffee align Nucleic Acids ??? *83*
Q: I do not want to compute the alignment *83*
Q: I would like to force some residues to be aligned *83*

Q: I would like to use structural alignments. *84*
Q: I want to build my own libraries. *84*
Q: I want to use my own tree *84*
Q: I want to align coding DNA *85*
Q: I do not want to use all the possible pairs when computing the library *85*
Q: I only want to use specific pairs to compute the library *85*
Q: There are duplicates or quasi-duplicates in my set *85*
Using Structures and Profiles **86**
Q: Can I align sequences to a profile with T-Coffee? *86*
Q: Can I align sequences Two or More Profiles? *86*
Q: Can I align two profiles according to the structures they contain? *86*
Q: T-Coffee becomes very slow when combining sequences and structures *86*
Q: Can I use a local installation of PDB? *87*
Alignment Evaluation **87**
Q: How good is my alignment? *87*
Q: What is that color index? *87*
Q: Can I evaluate alignments NOT produced with T-Coffee? *88*
Q: Can I Compare Two Alignments? *88*
Q: I am aligning sequences with long regions of very good overlap *88*
Q: Why is T-Coffee changing the names of my sequences!!!! *89*
Improving Your Alignment **89**
Q: How Can I Edit my Alignment Manually? *89*
Q: Have I Improved or Not my Alignment? *89*

Addresses and Contacts **90**
Contributors **90**
Addresses **90**

References **92**
T-Coffee **92**
Mocca **93**
CORE **94**
Other Contributions **94**
Bug Reports and Feedback **94**

Cheat Sheet: T-Coffee

Proteins

```
Mode                Command
=============================================================================
Very Fast           t_coffee sample_aln1.fasta -mode quickaln
                    lower -ndiag if the sequences are very similar
-----------------------------------------------------------------------------
Regular             t_coffee sample_aln1.fasta
                    use the output.html to estimate the MSA accuracy
-----------------------------------------------------------------------------
Very Accurate       t_coffee sample_aln1.fasta -mode accurate
                    slow, combines structures, sequences and profiles
-----------------------------------------------------------------------------
M-Coffee            t_coffee sample_aln1.fasta -mode mcoffee
                    combines most of the existing MSA packages
-----------------------------------------------------------------------------
3D-Coffee           t_coffee sample_aln1.fasta -mode 3dcoffee
                    uses the structure of your sequences if named with PDBID
-----------------------------------------------------------------------------
Expresso            t_coffee sample_aln1.fasta -mode expresso
                    finds structures homologous to your sequences
-----------------------------------------------------------------------------
PSI-Coffee          t_coffee sample_aln1.fasta -mode psicoffee
                    enriches your sequence with profile information
-----------------------------------------------------------------------------
```

DNA

```
R-Coffee            t_coffee three_cdna.fasta -mode cdna
```

RNA

```
Mode                Command
=============================================================================
R-Coffee            t_coffee sample_rnaseq1.fasta -mode rcoffee
                    use the predicted secondary structure of your sequences
-----------------------------------------------------------------------------
RM-Coffee           t_coffee sample_rnaseq1.fasta -mode rmcoffee
                    use M-Coffee + secondary structure prediction
-----------------------------------------------------------------------------
R-Coffee Consan     t_coffee sample_rnaseq1.fasta -mode rcoffee_consan
                    use rcoffee to combine consan alignments. Accurate and Slow
```

Memory Problems

memory	t_coffee sample_aln1.fasta -mode memory

Before You Start…

Foreword

A lot of the stuff presented here emanates form 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 callanques. 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.

This Document contains two main sections. The first one is a tutorial, where we go from simple things to more complicated and show you how to use all the subtleties of T-Coffee. We have tried to put as many of these functionalities on the web (www.tcoffee.org) but if you need to do something special and highly reproducible, the Command Line is the only way.

Pre-Requisite

This tutorial relies on the assumption that you have installed T-Coffee, version 6.18 or higher.

T-Coffee is a freeware open source running on all Unix-like platforms, including MAC-osX and Cygwin. All the relevant information for installing T-Coffee is contained in the Technical Documentation (tcoffee_technical.doc in the doc directory.)

T-Coffee cannot run on the Microsoft Windows shell. If you need to run T -Coffee on windows, start by installing cygwin (www.cygwin.com). Cygwin is a freeware open source that makes it possible to run a unix-like command line on your Microsoft Windows PC without having to reboot. Cygwin is free of charge and very easy to install. Yet, as the first installation requires downloading substantial amounts of data, you should make sure you have access to a broad-band connection.

In the course of this tutorial, we expect you to use a unix-like command line shell. If you work on Cygwin, this means clicking on the cygwin icon and typing commands in the window that appears. If you don't want to bother with command line stuff, try using the online tcoffee webserver at: **www.tcoffee.org**

Getting the Example Files of the Tutorial

We encourage you to try all the following examples with your own sequences/structures. If you want to try with ours, you can get the material from the example directory of the distribution. If you do not know where this file leaves or if you do not have access to it, the simplest thing to do is to:

1- download T-Coffee's latest version from www.tcoffee.org (Follow the link to the T-Coffee Home Page)

2- Download the latest distribution

3- gunzip <distrib>.tar.gz

4- tar -xvf <distrib>.tar

5- go into <distrib>/example

This is all you need to do to run ALL the examples provided in this tutorial.

What Is T-COFFEE ?

What is T-Coffee?

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

What does it do?

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

T-Coffee can also be used to compare alignments, reformat them or evaluate them using structural information, it is the mode known as seq_reformat.

What can it align?

T-Coffee will align nucleic and protein sequences alike. It will be able to use structural information for protein sequences with a known structure or the RNA sequences. On a new PC mid of the range, T-Coffee will align up to a 100 sequences, about 1000 amino acid long.

How can I use it?

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

```
PROMPT: t_coffee sample_seq1.fasta
```

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

Is There an Online Server

Yes, at www.tcoffee.org

Is T-Coffee different from ClustalW?

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

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…

If you want to align closely related sequences, T-Coffee can also be used in a fast mode, much faster than ClustalW, and about as accurate:

```
        PROMPT: t_coffee sample_seq1.fasta -mode quickaln
```

This mode works by only considering the best diagonals between two sequences. By default all the diagonals having a substitution score >0 are considered, but you can lower this by specifying:

```
        PROMPT: t_coffee sample_seq1.fasta -mode quickaln -ndiag=10
```

That will only consider the top 10 diagonals. This will be very useful if you have long and very similar sequences to align (DNA for instance).

Is T-Coffee very accurate?

T-Coffee combines methods, and can be made as accurate (and hopefully more) as the methods it combines. If you need a very accurate alignment (and you have the full package installed with SOAP);

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

If you cannot run this job, go to the first section of the technical manual (Installing BLAST for T-Coffee). You don't necessary need to install BLAST locally but you must have access to a remote server (EBI or NCBI).

This mode is very slow but also very accurate. On average this mode is about 10 % more accurate than normal aligners on sequences less than 30% similar. If you want something faster:

```
t_coffee sample_seq1.fasta
```

This is the normal mode. It is one of the most accurate of its kind, roughly like Probcons.

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

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

(NOT) Fetching Sequences

T-Coffee will NOT fetch sequences for you: you must select the sequences you want to align before hand. We suggest you use any BLAST server and format your sequences in FASTA so that T-COFFEE can use them easily. The expasy BLAST server (www.expasy.ch) provides a nice interface for integrating database searches.

Yet, the new modes of

Aligning Sequences

T-Coffee will compute (or at least try to compute!) accurate multiple alignments of DNA, RNA or Protein sequences.

Combining Alignments

T-Coffee allows you to combine results obtained with several alignment methods. For instance if you have an alignment coming from ClustalW, an other alignment coming from Dialign, and a structural alignment of some of your sequences, T-Coffee will combine all that information and produce a new multiple sequence alignment having the best agreement with all these methods (see the FAQ for more details)

```
PROMPT: t_coffee -aln=sproteases_small.cw_aln, sproteases_small.muscle,
sproteases_small.tc_aln -outfile=combined_aln.aln
```

Evaluating Alignments

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

```
PROMPT: t_coffee -infile=sproteases_small.aln -special_mode=evaluate
```

Combining Sequences and Structures

One of the latest improvements of T-Coffee is to let you combine sequences and structures, so that your alignments are of higher quality. You need to have sap package installed to fully benefit of this facility. If you have the EBI BLAST client installed (see installation procedure), you can run the following:

```
PROMPT: t_coffee 3d.fasta –special_mode=expresso
```

BLAST will identify the best PDB target for each sequences, and T-Coffee will use sap (or any other structural package) to align your structures and your sequences. If you do not have BLAST installed, or if you want to specify the templates yourself, you can use 3D-Coffee:

```
PROMPT: t_coffee 3d.fasta –special_mode=3dcoffee
```

In this case, the sequences must be names according to their PDB targets. All these network based operations are carried out using wget. If wget is not installed on your system, you can get it for free from (www.wget.org). To make sure wget is installed on your system, type

```
PROMPT: which wget
```

Identifying Occurrences of a Motif: Mocca

Mocca is a special mode of T-Coffee that allows you to extract a series of repeats from a single sequence or a set of sequences. In other words, if you know the coordinates of one copy of a repeat, you can extract all the other occurrences. If you want to use Mocca, simply type:

```
PROMPT: t_coffee –other_pg mocca sample_seq1.fasta
```

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

How Does T-Coffee works

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

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

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

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

and line 100 may say

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

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

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

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

Preparing Your Data: Reformatting and Trimming With seq_reformat

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 will realize that you have spend the whole day adding carriage return to each line in your files.

Seq_reformat

Accessing the T-Coffee Reformatting Utility

T-Coffee comes along with a very powerful reformatting utility named seq_reformat. You can use seq_reformat by invoking the t_coffee shell.

```
PROMPT: t_coffee -other_pg seq_reformat
```

This will output the online flag usage of seq_reformat. Seq_reformat recognizes automatically the most common formats. You can use it to:

Reformat your sequences.

extract sub-portions of alignments

Extract sequences.

In this section we give you a few examples of things you can do with seq_reformat:

Warning: after the flag -other_pg, the T-Coffee flags are not any more recognized. It is like if you were using a different programme

An overview of seq_reformat

seq_reformat is a reformatting utility. It reads in 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 modify your data, using any of the flag. If you want a complete list of things seq_reformat can do for you, try:

```
PROMPT: t_coffee -other_pg seq_reformat
```

Reformatting your data

Changing MSA formats

It can be necessary to change from one MSA format to another. If your sequences are in ClustalW format and you want to turn them into fasta, while keeping the gaps, try

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -output
fasta_aln > sproteases_small.fasta_aln
```

If you want to turn a clustalw alignment into an alignment having the pileup format (MSF), try:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -output msf
> sproteases_small.msf
```

Dealing with Non-automatically recognized formats

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:

```
    -in_f msf_aln for intance
```

Automated Sequence Edition

Removing the gaps from an alignment

If you want to recover your sequences from some pre-computed alignment, you can try:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -output
fasta_seq > sproteases_small.fasta
```

This will remove all the gaps.

Changing the case of your sequences

If you need to change the case of your sequences, you can use more sophisticated functions embedded in seq_reformat. We call these modifiers, and they are accessed via the -action flag. For instance, to write our sequences in lower case:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -action
+lower -output clustalw
```

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

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 color, you can put all the coordinates in a file, (one coordinate per line). Spans are not yet supported.

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_aln1.aln -action +upper
+edit_residue hmgb_chite 10 lower
```

Changing the case depending on the score

If you want to change the case depending on the score, you must either evaluate your alignment, or provide cache (see next section for the cache). If you want to evaluate on the fly, try:

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_aln1.aln -in3
sample_aln1.aln -action +upper +3evaluate idmat +lower '[5-9]'
```

Will lower the case of every residue identical to more than 50% of the residue in its column.

Protecting Important Sequence Names

Few programs support long sequence names. Sometimes, when going through some pipeline the names of your sequences can be damaged (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.

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. For instance:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_large.fasta -output
code_name > sproteases_large.code_name
```

Will create a file where each original name is associated with a coded name (Cxxxx). You can then use this file to either code or decode your dataset. For instance, the following command:

```
PROMPT: t_coffee -other_pg seq_reformat -code sproteases_large.code_name -in
sproteases_large.fasta >sproteases_large.coded.fasta
```

Will code all the names of the original data. You can work with the file sproteases_large.coded.fasta, and when you are done, you can de-code the names of your sequences using:

```
PROMPT: t_coffee -other_pg seq_reformat -decode sproteases_large.code_name -
in sproteases_large.coded.fasta
```

Colouring/Editing Residues in an Alignment

Coloring specific types of residues

You can color all the residues of your sequences on the fly. For instance, the following command:

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_aln1.aln -in3
sample_aln1.aln -action  +3convert a0 -output color_html > colored.html
```

will color all the As in color 0 (blue).

Coloring a specific residue of a specific sequence

If you want to color a specific residue, 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.

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_aln1.aln -action
+color_residue hmgb_chite 10 1 -output color_html > color.html
```

Coloring according to the conservation

Use the +evaluate flag if you want to color your alignment according to its conservation level

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_aln1.aln -in3
sample_aln1.aln -action +3evaluate pam250mt- output color_html > color.html
```

You can also use the boxshade scoreing scheme:

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_aln1.aln -in3
sample_aln1.aln -action +3evaluate boxshade -output color_html > color.html
```

Colouring/Editing residues in an Alignment Using a Cache

Overview

To color an alignment, two files are needed: the alignment (aln) and the cache (cache). The cache is a file where residues to be colored are declared along with the colors. Nine different colors are currently supported. They are set by default but can be modified by the user (see last changing default colors). The cache can either look like a standard sequence or alignment file (see below) or like a standard T-Coffee library (see next section). In this section we show you how to specifically modify your original sequences to turn them into a cache.

In the cache, the colors of each residue are declared with a number between 0 and 9. Undeclared residues will appear without any color in the final alignment.

Preparing a Sequence or Alignment Cache

Let us consider the following file:

```
CLUSTAL FORMAT

B               CTGAGA-AGCCGC---CTGAGG--TCG
C               TTAAGG-TCCAGA---TTGCGG--AGC
D               CTTCGT-AGTCGT---TTAAGA--ca-
A               CTCCGTgTCTAGGagtTTACGTggAGT
                 *  *      *     *  *
```

The command

```
PROMPT: t_coffee -other_pg seq_reformat -in=sample_aln6.aln -
output=clustalw_aln -out=cache.aln -action +convert 'Aa1' '.--' +convert '#0'
```

The conversion will proceed as follows:

-conv indicates the filters for character conversion:

- will remain -

A and a will be turned into 1

All the other symbols (#) will be turned into 0.

-action +convert, indicates the actions that must be carried out on the alignment before it is output into cache.

This command generates the following alignment (called a cache):

```
CLUSTAL FORMAT for SEQ_REFORMAT Version 1.00, CPU=0.00 sec, SCORE=0, Nseq=4, Len=27

B               000101-100000---000100--000
C               001100-000101---000000--100
D               000000-100000---001101--01-
A               000000000100100001000000100
```

Other alternative are possible. For instance, the following command:

```
PROMPT: t_coffee -other_pg seq_reformat -in=sample_aln6.aln -output=fasta_seq
-out=cache.seq -action +convert 'Aa1' '.--' +convert '#0'
```

will produce the following file cache_seq

```
>B
000101100000000100000
>C
001100000101000000100
>D
00000010000000110101
>A
000000000010010000100000100
```

where each residue has been replaced with a number according to what was specified by conv. Note that it is not necessary to replace EVERY residue with a code. For instance, the following file would also be suitable as a cache:

```
PROMPT: t_coffee -other_pg seq_reformat -in=sample_aln6.aln -output=fasta_seq
-out=cache -action +convert 'Aa1' '.--'
```

```
>B
CTG1G11GCCGCCTG1GGTCG
>C
TT11GGTCC1G1TTGCGG1GC
>D
CTTCGT1GTCGTTT11G1c1
>A
CTCCGTgTCT1GG1gtTT1CGTgg1GT
```

Preparing a Library Cache

The Library is a special format used by T-Coffee to declare special relationships between pairs of residues. The cache library format can also be used to declare the color of specific residues in an alignment. For instance, the following file

```
! TC_LIB_FORMAT_01
4
A 27 CTCCGTgTCTAGGagtTTACGTggAGT
B 21 CTGAGAAGCCGCCTGAGGTCG
C 21 TTAAGGTCCAGATTGCGGAGC
D 20 CTTCGTAGTCGTTTAAGAca
#1 1
    1     1   3
    4     4   5
#3 3
    6     6   1
    9     9   4
! CPU 240
! SEQ_1_TO_N
```

sample_lib5.tc_lib declares that residue 1 of sequence 3 will be receive color 6, while residue 20 of sequence 4 will receive color 20. 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 (c.f. technical doc, library format). For instance:

```
! TC_LIB_FORMAT_01
4
A 27 CTCCGTgTCTAGGagtTTACGTggAGT
B 21 CTGAGAAGCCGCCTGAGGTCG
C 21 TTAAGGTCCAGATTGCGGAGC
D 20 CTTCGTAGTCGTTTAAGAca
#1 1
    +BLOCK+ 10 1      1    3
    +BLOCK+ 5  15     15   5
#3 3
    6     6   1
    9     9   4
! CPU 240
! SEQ_1_TO_N
```

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.

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:

```
   PROMPT: t_coffee -other_pg seq_reformat -in=sample_aln6.aln -
struc_in=sample_aln6.cache -struc_in_f number_fasta -output=color_html -
out=x.html
```

This will produce a colored version readable with any standard web browser, while:

```
   PROMPT: t_coffee -other_pg seq_reformat -in=sample_aln6.aln -
struc_in=sample_aln6.cache -struc_in_f number_fasta -output=color_pdf -
out=x.pdf
```

This will produce a colored version readable with acrobat reader.

Warning: ps2pdf must be installed on your system

You can also use a cache library like the one shown above (sample_lib5.tc_lib):

```
PROMPT: t_coffee -other_pg seq_reformat -in=sample_aln6.aln -
struc_in=sample_lib5.tc_lib -output=color_html -out=x.html
```

Changing the default colors

Colors are hard coded in the program, but if you wish, you can change them, simply create a file named:

```
   seq_reformat.color
```

That is used to declare the color values:

```
0 #FFAA00 1 0.2 0
```

This indicates that the value 0 in the cache corresponds now to #FFAA00 in html, and in RGB 1, 0.2 and 0. 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**

Evaluating an alignment and producing a cache

Evaluating an alignment with T-Coffee

As suggested in a previous section, it is possible to evaluate the accuracy of any alignment using a T-Coffee library. The simplest way to do that is to compute a default library and evaluate the target alignment against this library:

```
PROMPT: t_coffee -infile sample_aln1.aln -mode evaluate
```

This command will output a file named sample_aln1.score_asccii that can then be used to either evaluate the local accuracy of the alignment or automatically filter it using the seq_reformat utility.

In some circumstances, you may also want to evaluate your alignment against a pre-computed library. This can be easily achieved:

```
PROMPT: t_coffee -infile sample_aln1.aln -out_lib sample_aln1.tc_lib -
lib_only
```

```
PROMPT: t_coffee -infile sample_aln1.aln -mode evaluate -lib
sample_aln1.tc_lib
```

When using this last command, the reference library will be the one provided by the user. The local score thus reported is the CORE index.

Evaluating the level of conservation with a substitution matrix

It is possible to use seq_reformat in a similar way to infer the local level of identity, either using an identity matrix or with any regular matrix, in which case, every residue with a substitution score higher than 0 is counted as an identity. This can be achieved as follows for identity measure:

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_aln1.aln -action +evaluate
idmat -output score_ascii
```

Or with the following for measuring similarity with a blosum62

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_aln1.aln -action +evaluate
blosum62mt -output score_ascii
```

Finally, it is also possible to display in color the conservation levels:

```
   PROMPT: t_coffee -other_pg seq_reformat -in sample_aln1.aln -action
+evaluate blosum62mt -output score_html > x.html
```

Selective Reformatting

Removing gapped columns

You can remove all the columns containing a certain proportion of gaps. For instance:

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_aln7.aln -action +rm_gap
50
```

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).

Selectively turn some residues to lower case

Consider the following alignment (sample_aln7.aln)

```
CLUSTAL  FORMAT  for  T-COFFEE  Version_4.62  [http://www.tcoffee.org],  CPU=0.04  sec,  SCORE=0,
Nseq=4, Len=28

A               CTCCGTGTCTAGGAGT-TTACGTGGAGT
B               CTGAGA----AGCCGCCTGAGGTCG---
D               CTTCGT----AGTCGT-TTAAGACA---
C               -TTAAGGTCC---AGATTGCGGAGC---
                 * ..        .*  * . *:
```

and the following cache (sample_aln7.cache_aln):

```
CLUSTAL  FORMAT  for  T-COFFEE  Version_4.62  [http://www.tcoffee.org],  CPU=0.04  sec,  SCORE=0,
Nseq=4, Len=28

A               3133212131022021-11032122021
B               312020----023323312022132---
D               311321----021321-11002030---
C               -110022133---020112322023---
```

You can turn to lower case all the residues having a score between 1 and 2:

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_aln7.aln -struc_in
sample_aln7.cache_aln -struc_in_f number_aln -action +lower '[1-2]'
```

```
CLUSTAL  FORMAT  for  T-COFFEE  Version_4.62  [http://www.tcoffee.org],  CPU=0.05  sec,  SCORE=0,
Nseq=4, Len=28

A               CtCCgtgtCtAggAgt-ttACgtggAgt
B               CtgAgA----AgCCgCCtgAggtCg---
D               CttCgt----AgtCgt-ttAAgACA---
C               -ttAAggtCC---AgAttgCggAgC---
                 * ..         .*   *  . *:
```

Note: that residues not concerned will keep their original case

Selectively modifying residues

The range operator is supported by three other important modifiers:

-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

-remove: to remove the residues within the range

-convert: to only convert the residues within the range.

For instance, to selectively turn all the G having a score between 1 and 2, use:

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_aln7.aln -struc_in
sample_aln7.cache_aln -struc_in_f number_aln -action +convert '[1-2]' CX
```

Keeping only the best portion of an alignment

To do this, you need an evaluation file that may have been generated with T-Coffee, either running a de-novo alignment

```
PROMPT: t_coffee sample_seq1.fasta -output score_ascii, aln
```

Or evaluating a pre-existing alignment

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_seq1.aln -action +evaluate
blosum62mt -output score_ascii
```

This generates a score_ascii file that you can then use to filter out the bad bits in your alignement:

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_seq1.aln -struc_in
sample_seq1.score_ascii -struc_in_f number_aln  -action +keep '[8-9]'
```

This command considers the individual score of each residue to trigger the filtering. It is also possible to do this according to the whole column. Simply add the "+use_cons" flag.

```
PROMPT: 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 '[8-
9]'
```

Extracting Portions of Dataset

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.

Extracting The High Scoring Blocks

It is possible to use a score_ascii file (as produced in the previous section) in order to extract high scoring portions of an alignment. For instance, the following command:

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

will keep all the residues having a column conservation score between 5 and 9

Note: Don't forget the simple quotes! (')

It is also possible to re-use pre-computed score_ascii files, such as those obtained when computing a T-Coffee multiple alignment. For instance, the following series of command will make it possible to extract the positions having a consistency score between 6 and 9:

```
   PROMPT: t_coffee sample_aln1.fasta -output score_ascii -outfile
sample1.score_ascii
```

```
  PROMPT: t_coffee -other_pg seq_reformat -in sample_aln1.aln -struc_in
sample1.score_ascii -struc_in_f number_aln -action +use_cons +keep '[8-9]'
```

Extracting Sequences According to a Pattern

You can extract any sequence by requesting a specific pattern to be found either in the name, the comment or the sequence. For instance, if you want to extract all the sequences whose name contain the word HUMAN:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -action
+grep NAME KEEP HUMAN -output clustalw
```

The modifier is "+grep". NAME indicates that the extraction is made according to the sequences names, and KEEP means that you will keep all the sequences containing the string HUMAN. If you wanted to remove all the sequences whose name contains the word HUMAN, you should have typed:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -action
+grep NAME REMOVE HUMAN -output clustalw
```

Note that HUMAN is case sensitive (Human, HUMAN and hUman will not yield the same results). You can also select the sequences according to some pattern found in their COMMENT section or directly in the sequence. For instance

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -action
+grep COMMENT KEEP sapiens -output clustalw
```

Will keep all the sequences containing the word ***sapiens*** in the comment section. Last but not least, you should know that the pattern can be any perl legal regular expression (See www.comp.leeds.ac.uk/Perl/matching.html for some background on regular expressions). For instance:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -action
+grep NAME REMOVE '[ILM]K' -output clustalw
```

Will extract all the sequences containing the pattern [ILM]K.

Extracting Sequences by Names

Extracting Two Sequences: If you want to extract several sequences, in order to make a subset. You can do the following:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -action
+extract_seq_list 'sp|P29786|TRY3_AEDAE' 'sp|P35037|TRY3_ANOGA'
```

Note the single quotes ('). They are meant to protect the name of your sequence and prevent the UNIX shell to interpret it like an instruction.

Removing Columns of Gaps. Removing intermediate sequences results in columns of gaps appearing here and there. Keeping them is convenient if some features are mapped on your alignment. On the other hand, if you want to remove these columns you can use:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -action
+extract_seq_list 'sp|P29786|TRY3_AEDAE' 'sp|P35037|TRY3_ANOGA' +rm_gap
```

Extracting Sub sequences: You may want to extract portions of your sequences. This is possible if you specify the coordinates after the sequences name:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -action
+extract_seq 'sp|P29786|TRY3_AEDAE' 20 200 'sp|P35037|TRY3_ANOGA' 10 150
+rm_gap
```

Keeping the original Sequence Names. Note that your sequences are now renamed according to the extraction coordinates. You can keep the original names by using the +keep_name modifier:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -action
+keep_name +extract_seq 'sp|P29786|TRY3_AEDAE' 20 200 'sp|P35037|TRY3_ANOGA'
10 150 +rm_gap
```

Note: +keep_name must come BEFORE +extract_seq

Removing Sequences by Names

Removing Two Sequences. If you want to remove several sequences, use rm_seq instead of keep_seq.

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -action
+remove_seq 'sp|P29786|TRY3_AEDAE' 'sp|P35037|TRY3_ANOGA'
```

Extracting Blocks Within Alignment

Extracting a Block. If you only want to keep one block in your alignment, use

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -action
+extract_block cons 150 200
```

In this command line, ***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. For instance:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -action
+extract_block 'sp|Q03238|GRAM_RAT' 10 200
```

Concatenating Alignments

If you have extracted several blocks and you now want to glue them together, you can use the cat_aln function

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -action
+extract_block cons 100 120 > block1.aln

PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small.aln -action
+extract_block cons 150 200 > block2.aln

PROMPT: 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.

Analyzing your Multiple Sequence Alignment

Estimating the diversity in your alignment

It is easy to measure the level of diversity within your multiple sequence alignment. The following command:

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_aln1.aln -output sim
```

Will output all the pairwise identities, as well as the average level of identity between each sequence and the others. You can sort and grep in order to select the sequences you are interested in.

Reducing and improving your dataset

Large datasets are problematic because they can be difficult to analyze. The problem is that when there are too many sequences, MSA programs tend to become very slow and inaccurate. Furthermore, you will find that large datasets are difficult to display and analyze. In short, the best size for an MSA dataset is between 20 and 40 sequences. This way you have enough sequences to ***see*** the effect of evolution, but at the same time the dataset is small enough so that you can visualize your alignment and recompute it as many times as needed.

Note: If your sequence dataset is very large, seq_reformat will compute the similarity matrix between your sequences once only. It will then keep it in its cache and re-use it any time you re-use that dataset. In short this means that it will take much longer to run the first time.

Extracting the N most informative sequences

To be informative, a sequence must contain information the other sequences do not contain. The N most informative sequences are the N sequences that are as different as possible to one another, given the initial dataset.

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_large.fasta -action
+trim _seq_n10  -output fasta_seq
```

The arguments 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 generaly more accurate to use unaligned sequences.

The argument ***_n10*** means you want to extract the 10 most informative sequences. If you would rather extract the 20% most informative sequences, use

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_large.fasta -action
+trim _seq_N20  -output fasta_seq
```

Extracting all the sequences less than X% identical

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, to generate a dataset where no pair of sequences has more than 50% identity, use:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_large.fasta -action
+trim _seq_%%50_
```

Speeding up the process

If you start form 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_

Note the difference of speed between these two command and the previous one:

```
PROMPT: t_coffee -other_pg seq_reformat -in kinases.aln -action +trim
_aln_%%50_
```

```
t_coffee -other_pg seq_reformat -in kinases.fasta -action +trim _seq_%%50_
```

Of course, using the MSA will mean that you rely on a more approximate estimation of sequence similarity.

Forcing Specific Sequences to be kept

Sometimes you want to trim while making sure specific important sequences remain in your dataset. You can do so by providing trim with a **string.** Trim will keep all the sequences whose name contains the string. For instance, if you want to force trim to keep all the sequences that contain the word HUMAN, no matter how similar they are to one another, you can run the following command:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_large.fasta -action
+trim _seq_%%50 HUMAN
```

When you give this command, the program will first make sure that all the HUMAN sequences are kept and it will then assemble your 50% dataset while keeping the HUMAN sequences. Note that string is a perl regular expression.

By default, string causes all the sequences whose name it matches to be kept. You can also make sure that sequences whose COMMENT or SEQUENCE matches string are kept. For instance, the following line

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_large.fasta -action
+trim _seq_%%50_fCOMMENT '.apiens'
```

Will cause all the sequences containing the regular expression '.apiens' in the comment to be kept. The _f symbol before COMMENT stands for "_field" If you want to make a selection on the sequences:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_large.fasta -action
+trim _seq_%%50_fSEQ '[MLV][RK]'
```

You can also specify the sequences you want to keep. To do so, give a fasta file containing the name of these sequences via the -in2 file

```
PROMPT:t_coffee -other_pg seq_reformat -in sproteases_large.fasta -in2
sproteases_small.fasta -action +trim _seq_%%40
```

Identifying and Removing Outlayers

Sequences that are too distantly related from the rest of the set will sometimes have very negative effects on the overall alignment. To prevent this, it is advisable not to use them. This can be done when trimming the sequences. For instance,

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_large.fasta  -action
+trim _seq_%%50_O40
```

The symbol _O stands for Outlayers. It will lead to the removal of all the sequences that have less than 40% average accuracy with all the other sequences in the dataset.

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, le us consider the two sequences:

sp|P21844|MCPT5_MOUSE   sp|P29786|TRY3_AEDAE

These sequences have 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:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_large.fasta -in2
sproteases_pair.fasta -action +chain > sproteases_chain.fasta
```

This will generate a dataset of 21 sequences, whith the following chain of similarity between your two sequences:

```
N: 21 Lower: 40 Sim: 25 DELTA: 15
#sp|P21844|MCPT5_MOUSE -->93 -->sp|P50339|MCPT3_RAT -->85 -->sp|P50341|MCPT2_MERUN -->72 --
>sp|P52195|MCPT1_PAPHA -->98 -->sp|P56435|MCPT1_MACFA -->97 -->sp|P23946|MCPT1_HUMAN -->8
1 -->sp|P21842|MCPT1_CANFA -->77 -->sp|P79204|MCPT2_SHEEP -->60 -->sp|P21812|MCPT4_MOUSE --
>90 -->sp|P09650|MCPT1_RAT -->83 -->sp|P50340|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|MCPT9_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_AEDAE
```

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.

Manipulating DNA sequences

Translating DNA sequences into Proteins

If your sequences are DNA coding sequences, it is always safer to align them as proteins.

Seq_reformat makes it easy for you to translate your sequences:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small_dna.fasta -
action +translate -output fasta_seq
```

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:

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small_dna.fasta -in2
sproteases_small.aln -action +thread_dna_on_prot_aln -output clustalw
```

Finding the Bona-Fide Sequences for the Back-Translation

Use the online server Protogene, available from www.tcoffee.org.

Guessing Your Back Translation

Back-translating means turning a protein sequence into a DNA sequence. If you do not have the original DNA sequence, this operation will not be exact, owing to the fact that the genetic code is degenerated. Yet, if a random-back translation is fine with you, you can use the following command.

```
PROMPT: t_coffee -other_pg seq_reformat -in sproteases_small_dna.fasta -in2
sproteases_small.aln -action +thread_dna_on_prot_aln  -output clustalw
```

In this process, codons are chosen randomly. For instance, if an amino-acid has four codons, the back-translation process will randomly select one of these. If you need more sophisticated back-translations that take into account the codon bias, we suggest you use more specific tools like: alpha.dmi.unict.it/~ctnyu/bbocushelp.html

Fetching 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, 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 www.rcsb.org). All these settings can be customized using environment variables (see the last section).

Fetching a PDB structure

If you want to fetch the chain E of the PDB structure 1PPG, you can use:

```
PROMPT: t_coffee -other_pg extract_from_pdb -infile 1PPGE
```

Fetching The Sequence of a PDB structure

To Fetch the sequence, use:

```
PROMPT: t_coffee -other_pg extract_from_pdb -infile 1PPGE -fasta
```

Will fetch the fasta sequence.

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:

```
export NO_REMOTE_PDB_FILE=1

or

t_coffee -other_pg extract_from_pdb -infile 1PPGE -fasta -no_remote_pdb_file
```

Manipulating RNA sequences with seq_reformat

Producing a Stockholm output: adding predicted secondary structures

Producing a consensus structure

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

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_rnaseq2.aln -action
+aln2alifold -output stockholm_aln
```

Adding a consensus structure to an alignment

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_rnaseq2.aln -action
+add_alifold -output stockholm_aln
```

Adding a pre-computed consensus structure to an alignment

The file sample_rnaseq2.aalifold contains the raw output of the alifold program captured as follows:

```
RNAalifold <sample_rnaseq2.aln  > sample_rnaseq2.alifold
```

It is possible to add this secondary structure to an alignment using:

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_rnaseq2.aln -in2
sample_rnaseq2.alifold -input2 alifold -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 stack it below the MSA.

It is also possible to stack Stockholm formatted secondary structures:

```
PROMPT: seq_reformat -in sample_rnaseq2.aln -in2 sample_rnaseq2.cons.stk -
action +add_alifold -output stockholm_aln
```

Analyzing an 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 re-computed by alifold.

Analyzing matching columns

+alifold2cov_stat will estimate the number of pairs of columns that are perfect Watson and Crick, those that are neutral (including a GU) and those that include correlated mutations. The WCcomp are the compensated mutations maintaining WC base pairing

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_rnaseq2.stk -action
+alifold2analyze stat
```

Other arguments can given, to display the list of paired positions and their status (compensated, Watson, etc)

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_rnaseq2.stk -action
+alifold2analyze list
```

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

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_rnaseq2.aln -action
+alifold2analyze aln
```

It is possible to turn this output into a colored one using:

```
PROMPT: t_coffee -other_pg seq_reformat -in sample_rnaseq2.aln -action
+alifold2analyze color_htm
```

Handling gapped columns

by default gapped column are ignored but they can be included by adding the tag usegap

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 folds can either be provided via Stockholm alignments

```
t_coffee -other_pg seq_reformat -in sample_rnaseq2.cw.stk -in2
sample_rnaseq2.tcoffee.stk -action +RNAfold_cmp
```

The top of the output (@@lines) summarizes the results that are displayed on the -in alignment. If the provided alignments do not have a fold, this fold will be estimated with alifold.

Manipulating Phylogenetic Trees with seq_reformat

Producing phylogenetic trees

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 UPGM or an NJ tree:

```
seq_reformat -in <aln> -action +aln2tree -output newick
```

Will use an identity matrix to compare your sequences and will output an unrooted NJ tree in newick format. If you want to produce a rooted UPGMA tree:

```
seq_reformat -in <aln> -action +aln2tree _TMODE_upgma -output newick
```

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:

```
   seq_reformat -in <aln> -action +aln2tree _MATRIX_sarmat -output newick
```

All these parameters can be concatenated:

```
   seq_reformat -in <aln> -action +aln2tree _TMODE_upgma_MATRIX_sarmat -
output newick
```

Bootstrap facilities will also be added at some point … For now we recommend you use Phylip if you need some serious phylogeny…

Comparing two phylogenetic trees

Consider the following file (sample_tree1.dnd)

```
(( A:0.50000, C:0.50000):0.00000,( D:0.00500, E:0.00500):0.99000, B:0.50000);
```

and the file sample_tree3.dnd.

```
(( E:0.50000, C:0.50000):0.00000,( A:0.00500, B:0.00500):0.99000, D:0.50000);
```

You can compare them using:

```
seq_reformat -in sample_tree2.dnd -in2 sample_tree3.dnd -action +tree_cmp -
output newick
```

```
tree_cpm|T: 75 W: 71.43 L: 50.50
tree_cpm|8 Nodes in T1 with 5 Sequences
tree_cmp|T: ratio of identical nodes
tree_cmp|W: ratio of identical nodes weighted with the min Nseq below node
tree_cmp|L: average branch length similarity
(( A:1.00000, C:1.00000):-2.00000,( D:1.00000, E:1.00000):-2.00000, B:1.00000);
```

Please consider the following aspects when exploiting these results:

-The comparison is made on the unrooted trees

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 (Number leaf left, Number leaf Right)

L: Fraction of branch length difference between the two considered trees.

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

Positive values (i.e. 2, 5) 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.

You can extract this tree for further usage by typing:

```
cat outfile | grep -v "tree_cmp"
```

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). The following command:

```
seq_reformat -in <aln> -in2 <reftree> -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> <blck_end> <block score> <block Length>
```

Pruning Phylogenetic Trees

Pruning removes leaves from an existing tree and recomputes distances so that no information is lost

Consider the file sample_tree2.dnd:

```
 (( A:0.50000, C:0.50000):0.00000,( D:0.00500, E:0.00500):0.99000, B:0.50000);
```

And the file sample_seq8.seq

```
>A
>B
>C
>D
```

Note: Sample_seq8 is merely a FASTA file where sequences can be omitted. Sequences can be omitted, but you can also leave them, at your entire convenience.

```
seq_reformat -in sample_tree2.dnd -in2 sample_seq8.seq -action +tree_prune -
output newick
```

```
 (( A:0.50000, C:0.50000):0.00000, B:0.50000, D:0.99500);
```

Building Multiple Sequence Alignments

How to generate The Alignment You Need?

What is a Good Alignment?

This is a trick question. A good alignment is an alignment that makes it possible to do good biology. If you want to reconstruct a phylogeny, a good alignment will be an alignment leading to an accurate reconstruction.

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 (within the reference datasets), all the reference collections agree with one another and all the established methods give roughly the same results. These alignments can be trusted blindly.

Below 40% accuracy within the reference datasets, 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 seem to indicate that consistency based methods (T-Coffee, Mafft-slow and ProbCons) have an edge over traditional methods.

When dealing with distantly related sequences, the only way to produce reliable alignments is to us 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*** doing 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.

-Secondly, the difference in accuracy (as measured on reference datasets) between all the available methods is not incredibly high. 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.

Last, but not least, bear in mind that these methods have only been evaluated by comparison with reference structure based sequence alignments. 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.

The Main Methods and their Scope

There are many MSA packages around. The main ones are ClustalW, Muscle, Mafft, T-Coffee and ProbCons. 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.

These packages offer a complex trade-off between speed, accuracy and versatility.

ClustalW: everywhere you look

ClustalW is still the most widely used multiple sequence alignment package. Yet things are gradually changing as recent 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.

Mafft and Muscle: Aligning Many Sequences

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 it is 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. It becomes much more accurate but also slower, and more sensitive to the number of sequences.

The alignments generated using the fast modes of these programs will be very suitable for several important applications such as:

-Distance based phylogenetic reconstruction (NJ trees)

-Secondary structure predictions

However they may not be suitable for more refined application such as

-Profile construction

-Structure Modeling

-3D structure prediction

-Function analysis

In that case you may need to use more accurate methods

T-Coffee and ProbCons: Slow and 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. Each library line is a constraint and the purpose is to assemble the alignment that accommodates the more all the constraints.

It is very much like building a high school schedule, where each teachers says something "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.

All these packages are ideal for the following applications:

-Profile reconstruction

-Function analysis

-3D Prediction

Choosing The Right Package

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 and it is 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 with T-Coffee.

	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 (T-Coffee Tutorial, C. Notredame, www.tcoffee.org). 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.

	MUSCLE	MAFFT	PROBCONS	T-COFFEE	CLUSTALW
Dist Based Phylogeny	+++	+++	++	++	++
ML or MP Phylogeny	++	+++	+++	+++	++
Profile Construction	++	+++	+++	+++	++
3D Modeling	++	++	++	+++	+
Secondary Structure P	+++	+++	++	++	++

Table 2. Most Suitable Appplications of each package (T-Coffee Tutorial, C. Notredame, www.tcoffee.org). 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.

Computing Multiple Sequence Alignments With T-Coffee

Computing Very accurate (but slow) alignments with PSI-Coffee

PSI-Coffee builds a profile associated with each of your input sequence and then makes a multiple profile alignment. If you do not have any structure, it is the most accurate mode of T-Coffee.

```
PROMPT: t_coffee sproteases_small.fasta -mode psicoffee
```

If you want to go further, and be even slower, you can use the accurate mode that will combine profile and structural information

```
PROMPT: t_coffee sproteases_small.fasta -mode accurate
```

It is probably one of the most accurate way of aligning sequences currently available.

A Simple Multiple Sequence Alignment

T-Coffee is meant to be run like ClustalW. This means you can use it like ClustalW for most simple applications. For instance, the following instruction

```
PROMPT: t_coffee sproteases_small.fasta
```

This instruction will compute a multiple sequence alignment of your sequences, using the default mode of T-Coffee. It will output the alignment on the screen and in a file named sproteases_small.aln. This file contains your alignment in ClustalW format.

The program will also output a file named sproteases_small.dnd that contains the guide tree used to assemble the progressive alignment.

Controlling the Output Format

If you need to, you can also trigger different ouput formats using the -output flag:

```
PROMPT: t_coffee sproteases_small.fasta -
output=clustalw,fasta_aln,msf
```

You can specify as many formats as you want.

Computing a Phylogenetic tree

T-Coffee is not a phylogeny package. Yet, it has some limited abilities to turn your MSA into a phylogenetic tree. This tree is a Neighbor Joining Phylogenetic tree, very similar to the one you could compute using ClustalW.

```
PROMPT: t_coffee sproteases_small.fasta -
output=clustalw,fasta_aln,msf
```

The phylogenetic tree is the file with the ph extension. Never use the .dnd tree in place of a genuine phylogenetic tree. The phylogenetic tree output by T-Coffee is only an indication. You should produce a bootstrapped phylogenetic tree using packages like Phylip (bioweb.pasteur.fr/seqanal/phylogeny/phylip-uk.html). You can visualize your tree using online tree drawing programs like phylodendron (iubio.bio.indiana.edu/treeapp/treeprint-form.html).

Using Several Datasets

If your sequences are spread across several datasets, you can give all the files via the -seq flag:

```
PROMPT: t_coffee -seq=sprotease1_small.fasta,sprotease2_small.aln
-output=clustalw,fasta_aln,msf
```

Note that you can give as many file as you want (the limit is 200) and that the files can be in any format. If you give an alignment, the gaps will be reset and your alignment will only provide sequences.

Sequences with the same name between two files are assumed to be the same sequence. If their sequences differ, they will be aligned and replaced by the consensus of that alignment. This process is known as sequence reconciliation.

You should make sure that there are no duplicates in your alignment, especially when providing multiple datasets.

How Good is Your Alignment

Later in this tutorial we show you how to estimate the accuracy of your alignment. Before we go into details, you should know that the number that comes on the first line of the header (in ClustalW format) is the score of your alignment.

CLUSTAL FORMAT for T-COFFEE Version_4.32 [http://www.tcoffee.org], CPU=19.06 sec, **SCORE=37**, Nseq=19, Len=341

You can use this value to compare alternative alignments of the same sequences. Alignments with a score higher than 40 are usually pretty good.

Doing it over the WWW

You can run T-Coffee online at www.tcoffee.org. Use the regular or the advanced form of the T-Coffee server.

Aligning Many Sequences

Aligning Very Large Datasets with Muscle

T-Coffee is not a good choice if you are dealing with very large datasets, use Mafft or Muscle. To align a large dataset with Muscle, try:

```
muscle -infile sproteases_large.fasta > sproteases_large.muscle
```

To use the fastest possible mode (less accurate) run:

```
muscle -in sproteases_large.fasta -maxiters 1 -diags -sv -
distance1 kbit20_3 > sproteases_large.muscle
```

Aligning Very Large Alignments with Mafft

The fastest mode with Mafft can be achieved using:

```
mafft --retree 2 input > output
```

Aligning Very Large Alignments with T-Coffee

T-Coffee is not very well gifted for aligning large datasets, but you can give it a try using a special option that generates approximate alignments. These alignments should roughly have the same accuracy as ClustalW. They are acceptable for sequences more than 40% identical.

```
PROMPT: t_coffee sproteases_large.fasta -mode quickaln
```

Shrinking Large Alignments With T-Coffee

Once you have generated your large alignment, you may nedd/want to shrink it to a smaller one, that will be (hopefuly) as informative and easier to manipulate. For that purpose, use the trim option (described in detail in the first section of this document).

```
PROMPT: t_coffee -other_pg seq_reformat -in
sproteases_large.muscle -action +trim _n20 -output >
sproteases_large_muscle_trim.aln
```

Modifying the default parameters of T-Coffee

The main parameters of T-Coffee are similar to those of ClustalW. They include the substitution matrix and the 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.

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, as described in the technical manual.

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. This may explain why theses 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:

```
PROMPT: t_coffee sproteases_small.fasta -matrix=blosum30mt -
outfile=b30.aln

PROMPT: t_coffee sproteases_small.fasta -matrix=blosum80mt -
outfile=b80.aln

PROMPT: t_coffee sproteases_small.fasta -matrix=pam350mt -outfile
p350.aln
```

You will get two alignments that have roughly the same score but are 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.

Comparing Two Alternative Alignments

If you change the parameters, you will end up with alternative alignemnts. 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:

```
PROMPT: t_coffee -other_pg aln_compare -al1 b30.aln -al2 p350.aln
```

This comparison will return the following result:

```
************************************************************
seq1       seq2          Sim   [ALL]           Tot
b30           19          32.6    93.7 [100.0]   [40444]
```

Where 93.7 is the percentage of similarity (sums of pairs) between the two alignments. It means that when considering every pair of aligned residues in b30 (40444), the program found that 93.7% of these pairs could be found in the alignment p350.aln.

Of course, this does not tell you where are the good bits, but you can get this information with the same program:

```
t_coffee -other_pg aln_compare -al1 b30.aln -al2 p350.aln -
output_aln -output_aln_threshold 50
```

```
sp|O35205|GRAK_MOUSE   M---r----fssw-------ALvslvagvym----------------SSECFHTEIIGGR
sp|Q7YRZ7|GRAA_BOVIN   M--ni----pfpf--sfppaIClllipgvfp----------------vs---cEGIIGGN
sp|P08884|GRAE_MOUSE   M--------ppv----------lilltlllp----------------l-GAGAEEIIGGH
sp|Q06606|GRZ2_RAT     M--------flf----------lfflvailp----------------v-NTEGGEIIWGT
sp|P21844|MCPT5_MOUSE  M---h----llt----------lhllllllg----------------s-STKAGEIIGGT
sp|P03953|CFAD_MOUSE   M---h----ssvy-------fvalvilgaav----------------CAAQPRGRILGGQ
sp|P00773|ELA1_RAT     M---l----rflv--F----ASlvlyghstq----------------DFPETNARVVGGA
sp|Q00871|CTRB1_PENVA  MIgkl----slll--V----CVavasgnpaagkpwhwKSPKPLVDPRIHVNATPRIVGGV
sp|P08246|ELNE_HUMAN   M--tlGR--rlac--L----FLacvlpalll----------------GGTALASEIVGGR
sp|P20160|CAP7_HUMAN   M--t-----rltv--L----ALlagllassr----------------AGSSPLLDIVGGR
sp|P80015|CAP7_PIG     -------------------------------------------------------IVGGR
sp|Q03238|GRAM_RAT     l-------------------LLllalktlwa----------------VGNRFEAQIIGGR
sp|P00757|KLKB4_MOUSE  M-----------w-------flilflalslggid-------------AAPP-----vqsq
sp|Q6H321|KLK2_HORSE   M-----------w-------flvlcldlslgetg-------------ALPPIQSRIIGGW
sp|Q91VE3|KLK7_MOUSE   M---------gvw-------llslitvllslale-------------tag-QGERIIDGY
sp|Q9Y5K2|KLK4_HUMAN   M-ataGN--pwgw-------flgylilgvag-sl-------------vsg-SCSQIINGE
sp|P29786|TRY3_AEDAE   M-------nqflfVSF---------calldsakvsaa------------tLSSGRIVGGF
sp|P35037|TRY3_ANOGA   M---iSNKiaillAVLvvav----acaqarvaqqhrsVQALPRFLPRPKYDVGHRIVGGF
sp|P07338|CTRB1_RAT    M--a------flwlvs---------cfalvgatfgcg---vptiqpv--LTGLSRIVNGE
                                                                               : .

sp|O35205|GRAK_MOUSE   EVQPHSRPFMASIQYR----SKHICGGVLIHPQWVLTAAHCYSWFprGHSPTVVLGAHSL
sp|Q7YRZ7|GRAA_BOVIN   EVAPHTRRYMALIK------GLKLCAGALIKENWVLTAAHCDlk----GNPQVILGAHST
sp|P08884|GRAE_MOUSE   VVKPHSRPYMAFVKSVDIEGNRRYCGGFLVQDDFVLTAAHCRN-----RTMTVTLGAHNI
sp|Q06606|GRZ2_RAT     ESKPHSRPYMAFIKFYDSNSEPHHCGGFLVAKDIVMTAAHCNG-----RNIKVTLGAHNI
sp|P21844|MCPT5_MOUSE  ECIPHSRPYMAYLEIVTSENYLSACSGFLIRRNFVLTAAHCAG-----RSITVLLGAHNK
sp|P03953|CFAD_MOUSE   EAAAHARPYMASVQVN----GTHVCGGTLLDEQWVLSAAHCMDGVtdDDSVQVLLGAHSL
sp|P00773|ELA1_RAT     EARRNSWPSQISLQYLSggswyHTCGGTLIRRNWVMTAAHCVSSQm---TFRVVVGDHNL
sp|Q00871|CTRB1_PENVA  EATPHSWPHQAALFId----DMYFCGGSLISSEWVLTAAHCMDGAg---FVEVVLGAHNI
sp|P08246|ELNE_HUMAN   RARPHAWPFMVSLQLr----GGHFCGATLIAPNFVMSAAHCVANVNV-RAVRVVLGAHNL
sp|P20160|CAP7_HUMAN   KARPRQFPFLASIQNq----GRHFCGGALIHARFVMTAASCFQSQNP-GVSTVVLGAYDL
sp|P80015|CAP7_PIG     RAQPQEFPFLASIQKq----GRPFCAGALVHPRFVLTAASCFRGKNS-GSASVVLGAYDL
sp|Q03238|GRAM_RAT     EAVPHSRPYMVSLQNT----KSHMCGGVLVHQKWVLTAAHCLSEP--LQQLKLVFGLHSL
sp|P00757|KLKB4_MOUSE  vdcENSQPWHVAVYRF----NKYQCGGVLLDRNWVLTAAHCYN-----DKYQVWLGKNNF
sp|Q6H321|KLK2_HORSE   ECEKHSKPWQVAVYHQ----GHFQCGGVLVHPQWVLTAAHCMS-----DDYQIWLGRHNL
sp|Q91VE3|KLK7_MOUSE   KCKEGSHPWQVALLKG----NQLHCGGVLVDKYWVLTAAHCKM-----GQYQVQLGSDKI
sp|Q9Y5K2|KLK4_HUMAN   DCSPHSQPWQAALVME----NELFCSGVLVHPQWVLSAAHCFQ-----NSYTIGLGLHSL
sp|P29786|TRY3_AEDAE   QIDIAEVPHQVSLQRS----GRHFCGGSIISPRWVLTRAHCTTNTDP-AAYTIRAGStd-
sp|P35037|TRY3_ANOGA   EIDVSETPYQVSLQYF----NSHRCGGSVLNSKWILTAAHCTVNLQP-SSLAVRLGSsr-
sp|P07338|CTRB1_RAT    DAIPGSWPWQVSLQDKt---gfHFCGGSLISEDWVVTAAHCGVKT----SDVVVAGEFDQ
                                   :           *.. ::    ::: * *           :  *
```

This is the alignment al1, but residues that have lost more than 50% of their pairing partner between the two alignments are now in lower case. In the section of this tutorial entitled comparing alignments, we show you more sophisticated ways to do this comparison.

For an even more drastic display, try:

```
t_coffee -other_pg aln_compare -al1 b30.aln -al2 p350.aln -
output_aln -output_aln_threshold 50 -output_aln_modif x
```

Changing Gap Penalties

Gap penalties are the core of the matter when it comes to multiple sequence alignments. 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 concistensy estimation (i.e. a 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:

```
PROMPT: t_coffee sproteases_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 into the core of the matter...

Two methods are used by default to build the library. One does global pairwise alignments and is named slow_pair, the other is named lalign_id_pair and and produces local alignments. These methods are specified via the -method flag. The default of this flag is:

```
PROMPT: t_coffee sproteases_small.fasta -
method=lalign_id_pair,slow_pair
```

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, you will need to do so explicitely. The default for lalign_id_pair is gop=-10, GEP=-4, MATRIX=blosum50mt. The default for slow_pair is: GOP=-10, GEP=-1 and MATRIX=blosum62mt. If you want to change this, try:

```
PROMPT: t_coffee sproteases_small.fasta -method
lalign_id_pair@EP@MATRIX@blosum62mt,slow_pair -outfile
sproteases_small.b62_aln
```

This means 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(default) to 50. We may assume this new alignment is more accurate than the previous one.

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).

Can You Guess The Optimal Parameters?

It is a trick question, but 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 add-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.

Using Many Methods at once

One of the most common situation when building multiple sequence alignments is to have several alignments produced by several alternative methods, and not knowing which one to choose. In this section, we show you that you can use M-Coffee to combine your many alignments into one single alignment. We show you here that you can either let T-Coffee compute all the multiple sequence alignments and combine them into one, or you can specify the methods you want to combine. M-Coffee is not always the best methods, 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 pre-computed alignments.

Using All the Methods at the Same Time: M-Coffee

In M-Coffee, M stands for Meta. To use M-Coffee, you will need several packages to be installed (see documentation). The following command:

```
PROMPT: t_coffee sproteases_small.fasta -mode mcoffee -output
clustalw, html
```

Will compute a Multiple Sequence Alignment with the following MSA packages: clustalw, poa, muscle, probcons, mafft, dialing-T, pcma and T-Coffee.

For those using debian, another mode is available

```
PROMPT: t_coffee sproteases_small.fasta -mode dmcoffee -output
clustalw, html
```

Will compute a Multiple Sequence Alignment with the following MSA packages: kalign, poa, muscle, probcons, mafft, dialing-T, and T-Coffee.

Package	Where From
ClustalW	can interact with t_coffee
Poa	http://www.bioinformatics.ucla.edu/poa/
Muscle	http://www.bioinformatics.ucla.edu/poa/
ProbCons	http://probcons.stanford.edu/
MAFFT	http://www.biophys.kyoto-u.ac.jp/~katoh/programs/align/mafft/
Dialign-T	http://dialign-t.gobics.de/
PCMA	ftp://iole.swmed.edu/pub/PCMA/
T-Coffee	www.tcoffee.org
Kalign	msa.cgb.ki.se/cgi-bin/msa.cgi
amap	bio.math.berkeley.edu/amap/

When this is done, all the alignments will be combined into one. If you open the file sproteases_small.html with your favorite web browser, you will see a colored version of your alignment.

The alignment is colored according to its consistency with all the MSA used to compute it. Regions in red have a high consistency and you can expect them to be fairly accurate. Regions in green/blue have the lowest consistency and you should not trust them.

Overall this alignment has a score of 80, which means that it is 80% consistent with the entire collection. This is a fairly high index, which means you can probably trust your alignment (at least where it is red).

Using Selected Methods to Compute your MSA

Using the 8 Methods of M-Coffee8 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. For instance, to combine MAFFT, Muscle, t_coffee and ProbCons, you can use:

```
PROMPT: t_coffee sproteases_small.fasta -
method=t_coffee_msa,mafft_msa,probcons_msa,muscle_msa -output=html
```

This will result in a computation where all the specified methods are mixed together

Combining pre-Computed Alignments

You may have a bunch of alignments that you have either pre-computed, or 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:

```
clustalw -infile=sproteases_small.fasta -gapopen=0 -outfile=g0.aln

clustalw -infile=sproteases_small.fasta -gapopen=-5 -
outfile=g5.aln

clustalw -infile=sproteases_small.fasta -gapopen=-10 -
outfile=g10.aln

clustalw -infile=sproteases_small.fasta -gapopen=-15 -
outfile=g15.aln
```

To combine them into ONE single alignment, use the -aln flag:

```
PROMPT: t_coffee sproteases_small.fasta -aln g0.aln g5.aln g10.aln
g15.aln -output clustalw html
```

As before, the score indicates a high level of consistency (91%) between all these alignments. This is an indication that the final alignment is probably correct.

Aligning Profiles

Sometimes, it is better to pre-align 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

Using Profiles as templates

Aligning One sequence to a Profile

Assuming you have a multiple alignment (sproteases_small.aln) here is a simple strategy to align one sequence to your profile:

```
PROMPT: t_coffee sproteases_oneseq.fasta  -profile
sproteases_small.aln
```

Aligning Many Sequences to a Profile

You can align as many sequences as you wish to your profile. Likewise, you can have as many profiles as you want. For instance, the following:

```
PROMPT: t_coffee sequences.fasta  -
profile=prf1.aln,prf2.aln,prf3.aln -outfile=combined_profiles.aln
```

Will make a multiple alignment of 3 profiles and 5 sequences. You can mix sequences and profiles in any proportion you like. You can also use all the methods you want although you should be aware that when using external methods (see the external method section in this tutorial), the profile is replaced with its consensus sequence, which will not be quite as accurate.

Methods supporting full profile information are: lalign_id_pair, slow_pair and proba_pair, clustalw_pair and clustalw_msa. All the other methods (internal or external) treat the profile as a consensus (less accurate).

Aligning Other Types of Sequences

Splicing variants

Splicing variants are especially challenging for most MSA programs. This is because the splicing variants need very long gaps to be inserted, while 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:

```
muscle -in sv.fasta -clw
```

You will quickly realise that your alignment is not very good and does not show where the alternative splicing coocurs. On the other hand, if you use T-Coffee, things become much clearer

```
PROMPT: t_coffee  sv.fasta
```

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:

```
PROMPT: t_coffee sv.fasta -method lalign_id_pair
```

Of course, the most distantly related your sequences, the harder the alignment of splicing variants

Aligning DNA sequences

Multiple Sequence Alignment methods are not at their best when aligning DNA sequences. Whenever you can, try using a local multiple sequence alignment 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...

```
PROMPT: 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. In practice, any time it encounters dna, the program will try to use "4dna" version of the requested methods. These methods have lower penalties and are better suited for dealing with nucleic acid sequences.

However, if you would rather use your own matrix, use:

```
PROMPT: t_coffee sample_dnaseq1.fasta –in
Mlalign_id_pair4dna@EP@MATRIX@idmat
```

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

Aligning RNA sequences

RNA sequences are very important and almost every-where these days. 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, smart people have thought about this. If you are interested in RNA, check: http://www.bio.inf.uni-jena.de/Software/MARNA/.

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

```
PROMPT: t_coffee three_cdna.fasta
```

You can immediately see that many gaps have sizes that are not multiple of 3 (codon size). Most of the information is lost. On the other hand, when using an appropriate alignment method that takes into account all the frames at the same time, we get something much more meaningful:

```
PROMPT: t_coffee three_cdna.fasta -method cdna_fast_pair
```

And most importantly, the frameshifts end up at the right place. You can even recover the corrected protein sequence using a special mode of seq_reformat:

```
PROMPT: t_coffee -other_pg seq_reformat -in three_cdna.aln -action
+clean_cdna +translate
```

+clean cdna is a small HMM that loops through each sequence and select the frame in order to maximize the similarity within the alignment.

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.

Single Sequence prediction

To make a secondary structure prediction with GOR, run the following. In this command line SSP is a hard coded mode. It prompts the computation of predicted secondary structures.

```
t_coffee sample_aln.fasta -template_file SSP
```

The predictions are then displayed in the files:

```
#### File Type= Template Protein Secondary Structure Format=  fasta_seq Name=
hmgb_chite.ssp
#### File Type= Template Protein Secondary Structure Format=  fasta_seq Name=
hmgl_trybr.ssp
#### File Type= Template Protein Secondary Structure Format=  fasta_seq Name=
hmgl_trybr3.ssp
#### File Type= Template Protein Secondary Structure Format=  fasta_seq Name=
hmgl_wheat.ssp
#### File Type= Template Protein Secondary Structure Format=  fasta_seq Name=
hmgl_wheat2.ssp
#### File Type= Template Protein Secondary Structure Format=  fasta_seq Name=
hmgt_mouse.ssp
```

Transmembrane structures can be carried out with:

```
t_coffee sample_aln.fasta -template_file TM
```

Multiple Sequence Predictions

Used this way, the method will produce for each sequence a secondary prediction file. GOR is a single sequence with a relatively low accuracy. It is possible to increase the accuracy by coupling BLAST and GOR, this can be achieved with the following command:

```
t_coffee sample_aln.fasta -template_file PSISSP
```

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.

A similar output can be obtained for Transmembrane segment predictions:

```
t_coffee sample_aln.fasta -template_file PSITM
```

Incorporation of the prediction in the alignment

It is possible to use the secondary prediction in order to reward the alignment of similar elements

```
t_coffee sample_aln.fasta -template_file PSISSP -
method_evaluate_mode ssp -method lalign_id_pair slow_pair
```

Likewise, it is possible to use this information with trans-membrane domains

```
t_coffee sample_aln.fasta -template_file PSITM -
method_evaluate_mode tm -method lalign_id_pair slow_pair
```

The overall effect is very crude and amounts to over-weighting 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.

Using other secondary structure predictions

If you have your own predictions, you can use them. All you need is to produce a template file where the file containing the secondary structure prediction is declared along with the sequence:

```
>hmgl_wheat _E_ hmgl_wheat.ssp
>hmgb_chite _E_ hmgb_chite.ssp
>hmgl_trybr3 _E_ hmgl_trybr3.ssp
>hmgl_wheat2 _E_ hmgl_wheat2.ssp
>hmgt_mouse _E_ hmgt_mouse.ssp
>hmgl_trybr _E_ hmgl_trybr.ssp
```

where each template looks like this:

```
>hmgl_wheat
CCCCCCCCCCCCHHHHHHHCCCCCCCCCHHHHHHHHHHHHHHHCCCCHHHHHHHHHHHHHHHHCE
```

You can then run T-Coffee using your own template file

```
t_coffee sample_aln.fasta -template_file <template_file> -
method_evaluate_mode ssp -method lalign_id_pair slow_pair
```

Output of the prediction

You can output a color coded version of your alignment using the predicted structures

```
t_coffee sample_aln.fasta -template_file PSISSP -output sec_html
```

A similar result can be obtained with trans-membrane regions:

```
t_coffee sample_aln.fasta -template_file PSITM -output tm_html
```

Combining Sequences and 3D-Structures

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 so long that their sequences have evolved beyond recognition. Yet, when assembling the correct 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.

If you are in a Hurry: Expresso

What is Expresso?

Expresso is the latest T-Coffee mode. It is not yet available for local installation, but you can run it from the www.tcoffee.org server. The principle of Expresso is simple: the server runs a BLAST between every sequence in your query against the PDB database. If it finds a structure similar enough to a sequence in your dataset (>60% identity), it will use that structure as a template for your sequence.

Template files look something like:

```
>sp|P08246|ELNE_HUMAN _P_ 1PPGE
>sp|P20160|CAP7_HUMAN _P_ 1AE5
>sp|P00757|KLKB4_MOUSE _P_ 1SGFX
>sp|Q6H321|KLK2_HORSE _P_ 1GVZA
>sp|P00773|ELA1_RAT _P_ 2D26C
>sp|Q00871|CTRB1_PENVA _P_ 1AZZB
>sp|P21844|MCPT5_MOUSE _P_ 1NN6A
>sp|O35205|GRAK_MOUSE _P_ 1MZDA
>sp|P07338|CTRB1_RAT _P_ 2CGAB
>sp|P80015|CAP7_PIG _P_ 1FY3A
>sp|P03953|CFAD_MOUSE _P_ 1FDPD
>sp|Q7YRZ7|GRAA_BOVIN _P_ 1OP8F
>sp|Q06606|GRZ2_RAT _P_ 1EUFA
>sp|P08884|GRAE_MOUSE _P_ 1FI8B
```

In a template file, **_P_** indicates that the template is of type structure (P for PDB). Template files can be generated manually or automatically by the Expresso server. Whenever possible t_coffee will then align your sequences using the structural information contained in the templates. If it encounters enough structures (as shown here) it will produce a genuine structure based sequence alignment.

Using Expresso

```
PROMPT: t_coffee three_pdb_two_seq.fasta -method
sap_pair,slow_pair -template_file PDB
```

Aligning Sequences and Structures

Mixing Sequences and Structures

Gather your sequences in the same file. Name your structures according to their PDB identifier. The file three_pdb_two_seq.fasta contains five sequences, three are the sequences of PDB structures and two are regular sequences.

What you want to do is to build a T-Coffee library where sequences with a known structures are aligned with a structure alignment program (like sap) while the other sequences are aligned using regular T-Coffee methods. You can achieve this with the following command:

```
PROMPT: t_coffee three_pdb_two_seq.fasta -method
sap_pair,slow_pair -template_file PDB
```

The option -template_file is here to tell the program how to find the PDB. In that case. EXPRESSO means that a remote BLAST (the EBI BLAST) will be used to identify the best targets. If your sequences are already named according to their PDB name, you can use:

```
 PROMPT: t_coffee three_pdb_two_seq.fasta -method
sap_pair,slow_pair -template_file _SELF_P_
```

SELF means that the PDB identifier is the name of the sequences, while _P_ is an indication that the template is indeed a PDB. These indications are necessary for T-Coffee to fetch the relevant structures.

The good news is that you do not need to have PDB installed locally as T-Coffee will automatically fetch the structures directly from RCSB (the home of PDB). Of course, if your dataset only contains structures, your alignment becomes a structural alignment.

If you have a fugue license, you can also add the fugue method to your run. Fugue will align the structures with sequences whose structure is unknown (this is called threading).

```
PROMPT: t_coffee three_pdb_two_seq.fasta -method
sap_pair,slow_pair,fugue_pair -template_file _SELF_P_
```

This can be written more concisely, using one of T-Coffee special_modes:

```
PROMPT: t_coffee three_pdb_two_seq.fasta -mode 3dcoffee
```

or

```
PROMPT: t_coffee three_pdb_two_seq.fasta -mode expresso
```

Using Sequences only

What often happens is that you have already built a dataset with sequences that are very similar to PDB sequences but not exactly identical. It may even be the case that the real sequence and the PDB one do not match exactly because of some genetic engineering on the structure. In this case, you have no structure whose sequence is exactly similar to the sequences in your dataset. All you need to do is to declare the equivalence sequences/structures and run T-Coffee, just like Expresso does.

The first step is to fill up a template file that contains an explicit declaration of the structures corresponding to your sequences. The format is very simple and fasta-like. You can use the file: sproteases_small.template_file

```
>sp|P08246|ELNE_HUMAN _P_ 1PPGE
>sp|P20160|CAP7_HUMAN _P_ 1AE5
>sp|P00757|KLKB4_MOUSE _P_ 1SGFX
>sp|Q6H321|KLK2_HORSE _P_ 1GVZA
```

In this file, the first line is telling us that sequence sp|P08246|ELNE_HUMAN is associated with the structural template 1PPGE. The sequence and the structure do not need to be identical although we recommend using structural templates more than 60% identical with your actual sequences (i.e. similar enough so that they generate a non ambiguous alignment). If your template file is ready, all you need to do is run the following command.

```
PROMPT: t_coffee sproteases_small.fasta -method slow_pair,
lalign_id_pair, sap_pair -template_file
sproteases_small.template_file
```

When you run this once, T-Coffee goes and fetches the structures. It will then align them using sap. It takes a lot of time to fetch structures, and it takes even more time to align them with sap. This is why T-Coffee saves these important intermediate results in a special location called the cache. By default, your cache is in ~/.t_coffee/cache, it is a good idea to empty it from time to time…

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. We have prepared two such profiles (prf1_pdb1.aln, prf2_pdb2.aln). You have two choices here. 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:

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

How Good Is Your Alignment ?

There are three strategies for evaluating your alignment. Structure is a killer. If you have two structures available for your protein family, you are in an ideal situation and you can use the iRMSD. If you don't, you are left with the option of using sequence based methods like the CORE index. These do pretty well in the CORE regions, but can be limited in the loops. Another killer, less often at hand, is the use of functional information. If you know some residues MUST be aligned because they are functionally related, you can easily set up an evaluation procedure using T-Coffee.

Evaluating Alignments with The CORE index

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. However, if you want to go further and estimate the local concistency (known as the CORE index). Simply request one extra output:

```
PROMPT: t_coffee sproteases_small.fasta -output=clustalw,html
```

The format html leads to the output of a file named sproteases_small.html. Open this file. It contains a colorized version of your alignment. In this colorized version, positions that have no concistency with the library are in blue, a little in green, better positions in yellow, then orange, then red. You can expect yellow positions to be entirely correct.

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:

```
PROMPT: t_coffee -infile=sproteases_small.g10.cw_aln -output=html
-score
```

Filtering Bad Residues

The local consistency score is between 0 and 9. If you need to build a profile or identify a signature or do some phylogeny, it may be a good idea to remove portions that are too unreliable. Here is how you can do it.

You will first need to produce a score version of your alignment in machine readable format. This format is called score_ascii:

```
PROMPT: t_coffee -infile=sproteases_small.g10.cw_aln -
output=score_ascii
```

You will then need to use seq_reformat to filter out the portions you are not interested in, using the file sproteases_small.g10.score_ascii as a cache for filtering. For instance, use the following command to only keep the residues whose score is between 5 and 9.

```
PROMPT: t_coffee -other_pg seq_reformat -in
sproteases_small.g10.cw_aln -struc_in
sproteases_small.g10.score_ascii -struc_in_f number_aln -action
+keep '[5-9]'
```

Not so neat... The reason is that most columns tend to be heterogenous and contain a couple of unhappy residues. If you would rather generate nice blocks, filter according to the consencus. This is easy and simply requires adding the +use_cons filter

```
PROMPT: t_coffee -other_pg seq_reformat -in
sproteases_small.g10.cw_aln -struc_in
sproteases_small.g10.score_ascii -struc_in_f number_aln -action
+use_cons +keep '[5-9]'
```

Removing columns of gaps is just as easy. You simply need to add the switch +rm_gap

```
PROMPT: t_coffee -other_pg seq_reformat -in
sproteases_small.g10.cw_aln -struc_in
sproteases_small.g10.score_ascii -struc_in_f number_aln -action
+use_cons +keep '[5-9]' +rm_gap
```

Filtering Gap Columns

You may want to remove columns that contain too many gaps. It is just a small variation around the rm_gap switch:

```
PROMPT: t_coffee -other_pg seq_reformat -in
sproteases_small.g10.cw_aln -struc_in
sproteases_small.g10.score_ascii -struc_in_f number_aln -action
+rm_gap 50
```

Will remove all the columns containing 40% or more gaps.

Evaluating an Alignment Using Structural Information: APDB and iRMSD

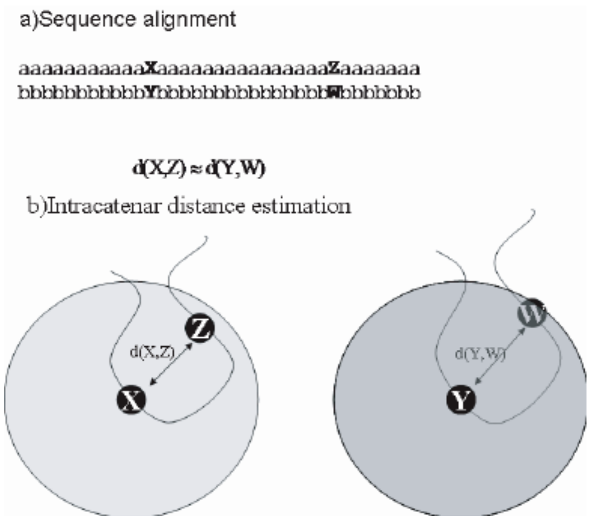
What is the iRMSD?

APDB and the iRMSD are two closely related measures meant to evaluate the accuracy of a sequence alignment 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 (It stands for intra-catener) 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 sequence alignment, one does not need to superpose the two structures and deduce a sequence alignment that will then be compared with the target alignment. In practice, we use a Normalized version of the iRMSD, the NiRMSD that makes it possible to compare alternative alignments of different length. From a structural point of view, the iRMSD has a meaning very similar to the iRMSD and it beahaves in a similar fashion from a numerical point of view (similar ranges in Angstroms).

The first step of APDB is to measure the distances between the Cα of each residue and its neighbors. Neighborhood is defined as a sphere of **radius –maximum_distance** (10Å by default). However, by setting **–local_mode** to “window”, the sphere can be replaced with a window of 1/2 size **'-maximum_distance'** residues.


a)Sequence alignment
aaaaaaaaaaaXaaaaaaaaaaaaaaaZaaaaaaa
bbbbbbbbbbbYbbbbbbbbbbbbbbbWbbbbbbb
d(X,Z) ≈ d(Y,W)
b)Intracatenar distance estimation
Z
X
d(X,Z)
W
Y
d(Y,W)

Given two aligned residues (X and Y on the Figure) the iRMSD measure 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 on Figure 1). The iRMSD is obtained by measuring the average Root Mean Square of these differences of distances. The lower the iRMSD, the better the alignment. However, an alignment can obtain a good iRMSD by simply having few aligned residues. To avoid this, the program also reports the NiRMSD= MIN(L1,L2)*iRMSD/Number Considered columns.

How to Efficiently Use Structural Information

When it comes to evaluating Multiple Sequence Alignments, nothing beats structural information. To use the methods we describe here, you will need to have at least two structures, similar enough (>60%) to two sequences in your dataset.

Here an outline of the best way to proceed:

1- Make sure you include two structures whose sequences are so distantly related that most of the other sequences are intermediates.

2- Align your sequences without using the structural information (i.e. t_coffee, muscle...)

3- Evaluate your alignment with iRMSD (see later in this section). The score will be S1

4- Realign your sequences, but this time using structural information (Expresso)

5- Measure the score of that alignment (Score=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 guaranty that the alignment of the intermediate sequences has improved as well, although in practice it often does

Evaluating an Alignment With the iRMSD Package

Let us evaluate the alignment produced by Expresso, using the template_file returned by expresso:

```
PROMPT: t_coffee -other_pg irmsd sproteases_small.expresso -
template_file sproteases_small.template_file
```

This will deliver a long output. The most interesting bit is at the bottom:

```
#TOTAL for the Full MSA
        TOTAL     EVALUATED:  52.90 %
        TOTAL     APDB:       81.59 %
        TOTAL     iRMSD:       0.71 Angs
        TOTAL     NiRMSD:      1.33 Angs
```

APDB is an older measure, less robust than the iRMSD and 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, the lower the NiRMSD, the better the alignment.

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:

Method	File	NiRMSD
Expresso	sproteases_small.expresso	1.33 Å
T-Coffee	sproteases_small.tc_aln	1.35 Å
ClustalW	sproteases_small.cw_aln	1.52 Å

Mafft	sproteases_small.mafft	1.36 Å
Muscle	sproteases_small.muscle	1.34 Å

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. Muscle seems to have a small edge here although the reality is that all these figures are impossible to distinguish with the notable exception of ClustalW

Identifying the most distantly related sequences in your dataset

In order to identify the most distantly related sequences in a dataset, you can use the seq_reformat utility, in order to compare all the sequences two by two and pick up the two having the lowest level of identity:

```
PROMPT: t_coffee -other_pg seq_reformat sproteases_small.fasta -
output sim_idscore | grep TOP |sort -rnk3
```

This sim_idscore indicates that every pair of sequences will need to be aligned when estimating the similarity. The ouput (below) indicates that the two sequences having the lowest level of identity are AEDAE and MOUSE. It may not be a bad idea to choose these sequences (if possible) for evaluating your MSA.

```
…
TOP        16   10       28.00  sp|P29786|TRY3_AEDAE      sp|Q6H321|KLK2_HORSE   28.00
TOP        16    7       28.00  sp|P29786|TRY3_AEDAE      sp|P08246|ELNE_HUMAN   28.00
TOP        16    1       28.00  sp|P29786|TRY3_AEDAE      sp|P08884|GRAE_MOUSE   28.00
TOP        15   14       27.00    sp|P80015|CAP7_PIG     sp|P00757|KLKB4_MOUSE   27.00
TOP        12    9       27.00  sp|P20160|CAP7_HUMAN      sp|Q91VE3|KLK7_MOUSE   27.00
TOP         9    7       27.00  sp|Q91VE3|KLK7_MOUSE      sp|P08246|ELNE_HUMAN   27.00
TOP        16    2       26.00  sp|P29786|TRY3_AEDAE     sp|P21844|MCPT5_MOUSE   26.00
```

Evaluating an Alignment according to your own Criterion

Establishing 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 Swissprot annotation of the two most 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, the first Histidine is at position 68, while in P21844 this Histidine is on position 66. We can therefore build a library that will check whether these residues are properly aligned in any MSA. The library will look like this:

```
! TC_LIB_FORMAT_01
2
sp|P21844|MCPT5_MOUSE                                                                        247
MHLLTLHLLLLLLLGSSTKAGEIIGGTECIPHSRPYMAYLEIVTSENYLSACSGFLIRRNFVLTAAHCAGRSITVLLGAHNKTSKED
TWQKLEVEKQFLHPKYDENLVVHDIMLLKLKEKAKLTLGVGTLPLSANFNFIPPGRMCRAVGWGRTNV
NEPASDTLQEVKMRLQEPQACKHFTSFRHNSQLCVGNPKKMQNVYKGDSGGPLLCAGIAQGIASYVHRNAKPPAVFTRISHYRPWI
NKILREN
sp|P29786|TRY3_AEDAE                                                                         254
MNQFLFVSFCALLDSAKVSAATLSSGRIVGGFQIDIAEVPHQVSLQRSGRHFCGGSIISPRWVLTRAHCTTNTDPAAYTIRAGSTD
RTNGGIIVKVKSVIPHPQYNGDTYNYDFSLLELDESIGFSRSIEAIALPDASETVADGAMCTVSGWGDT
KNVFEMNTLLRAVNVPSYNQAECAAALVNVVPVTEQMICAGYAAGGKDSCQGDSGGPLVSGDKLVGVVSWGKGCALPNLPGVYARV
STVRQWIREVSEV
#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 as a library to measure the concistency between your alignment and the correspondances declared in your library. The following command line also makes it possible to visualy display the agreement between your sequences and the library.

```
PROMPT: t_coffee -infile sproteases_small.aln -lib
charge_relay_lib.tc_lib  -score -output html
```

Integrating External Methods In T-Coffee

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 favourite method to be integrated, you will need to bite the bullet …

What Are The Methods Already Integrated in T-Coffee

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

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

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

List of INTERNAL Methods

Built in methods methods can be requested using the following names. To

proba_pair	*Adpated from Probcons, this method* ***[the current default]*** *uses a pair HMM to compute a pairwise alignment with a bi-phasic 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	
islow_pair	*Makes a global fasta alignmnet using the previously computed pairs as a library. `i` stands for iterative. Each pair of residue is given a score function of the weighting mode defined by -weight. The Library used for the computation is the one computed before the method is used. The resullt is therefore dependant on the order in methods and library are set via the –in flag.*
align_pdb_pair	*Uses the align_pdb routine to align two structures. The pairwise scores are those returnes by the align_pdb program. If a structure is missing, fast_pair is used instead. Each pair of residue is given a score function defined by align_pdb. [UNSUPORTED]*
lalign_id_pair	*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.*
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 amino-acid similarity and uses different penalties for amino-acid insertions and frameshifts. This alignment is turned into a library where matched nucleotides receive a score equql to the average level of identity at the amino-acid level. This mode is intended to clean cDNA obtained from ESTs, or to align pseudo-genes.*

WARNING: This method is currently unsuported.

PLUG-INs:List OF EXTERNAL METHODS

Plug-In: Using Methods Integrated in T-Coffee

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

```
Package              Where From
==========================================================
ClustalW             can interact with t_coffee
----------------------------------------------------------
Poa                  http://www.bioinformatics.ucla.edu/poa/
----------------------------------------------------------
Muscle        http://www.bioinformatics.ucla.edu/poa/
----------------------------------------------------------
ProbCons             http://probcons.stanford.edu/
----------------------------------------------------------
MAFFT  http://www.biophys.kyoto-u.ac.jp/~katoh/programs/align/mafft/
----------------------------------------------------------
Dialign-T            http://dialign-t.gobics.de/
----------------------------------------------------------
PCMA                 ftp://iole.swmed.edu/pub/PCMA/
----------------------------------------------------------
sap                  structure/structure comparisons
(obtain it from W. Taylor, NIMR-MRC).
---------------------------------------------------
Blast                www.ncbi.nih.nlm.gov
---------------------------------------------------
Fugue                protein to structure alignment program
                     http://www-cryst.bioc.cam.ac.uk/fugue/download.html
```

Once installed, most of these methods can be used as either pairwise or multiple alignment methods. Note that all these methods use Blosum62 as a default.

clustalw_pair	*Uses clustalw (default parameters) to align two sequences. Each pair of residue is given a score function of the weighting mode defined by -weight.*
clustalw_msa	*Makes a multiple alignment using ClustalW and adds it to the library. Each pair of residue is given a score function of the weighting mode defined by -weight.*
probcons_pair	*Probcons package: probcons.stanford.edu/.*
probcons_msa	*idem.*
muscle_pair	*Muscle package www.drive5.com/muscle/ .*
muscle_msa	*idem.*
mafft_pair	*www.biophys.kyoto-u.ac.jp/~katoh/programs/align/mafft/ .*
mafft_msa	*idem.*
pcma_msa	*pcma package*
pcma_pair	*pcma package*
poa_msa	*poa package*
poa_pair	*poa package*
dialignt_pair	*dialignt package*
dialignt_msa	*pcma package*
sap_pair	*Uses sap to align two structures. Each pair of residue is given a score function defined by sap. You must have sap installed on your system to use this method.*
fugue_pair	*Uses a standard fugue installation to make a sequence /structure alignment. Fugue installation must be standard. It does not have to include all the fugue packages but only:*

1- joy, melody, fugueali, sstruc, hbond

2-copy fugue/classdef.dat   /data/fugue/SUBST/classdef.dat

OR

Setenv MELODY_CLASSDEF=<location>

Setenv MELODY_SUBST=fugue/allmat.dat

All the configuration files must be in the right location.

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

```
PROMPT: t_coffee -seq sample_seq1.fasta -method
fast_pair,lalign_id_pair
```

Modifying the parameters of Internal and External Methods

Internal Methods

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

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

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

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:

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

@EP is here to indicate that you will pass an extra parameter

@PARAM1 is the name of this parameter

the next filed is the parameter itself, where:

%s replace spaces

%e replaces the equal sign

Of course, you must know the command line of the program you are trying to modify (clustalw in this case).

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.

Direct access to external methods

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

```
PROMPT: t_coffee sample_seq1.fasta –method=em@clustalw@pairwise
```

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

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

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

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

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

Customizing an external method (with parameters) for T-Coffee

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

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

```
PROMPT: t_coffee –other_pg unpack_clustalw_method.tc_method

PROMPT: t_coffee –other_pg unpack_generic_method.tc_method
```

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

```
*TC_METHOD_FORMAT_01
***************clustalw_method.tc_method*********
EXECUTABLE   clustalw
ALN_MODE            pairwise
IN_FLAG             -INFILE=
OUT_FLAG            -OUTFILE=
OUT_MODE            aln
PARAM        -gapopen=-10
SEQ_TYPE            S
*************************************************
```

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

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

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

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

```
PROMPT: t_coffee sample_seq1.fasta –method
clustalw_method.tc_method
```

Managing a collection of method files

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

Advanced Method Integration

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

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

When you run this method:

```
PROMPT: t_coffee –other_pg unpack_generic_method.tc_method

PROMPT: 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 this case –method clustalw indicates that the script should run clustalw on your data. The script tc_generic_method.pl is incorporated in t_coffee. Over the time, this script will be the place where novel methods will be integrated

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

```
PROMPT: t_coffee –other_pg=unpack_tc_generic_method.pl
```

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

The Mother of All method files…

```
*TC_METHOD_FORMAT_01
******************generic_method.tc_method*************
*
*       Incorporating new methods in T-Coffee
*       Cedric Notredame 17/04/05
*
*******************************************************
*This file is a method file
*Copy it and adapt it to your need so that the method
*you want to use can be incorporated within T-Coffee
*******************************************************
*                  USAGE                              *
*******************************************************
*This file is passed to t_coffee via -in:
*
*      t_coffee -in Mgeneric_method.method
*
*      The method is passed to the shell using the following
*call:
*<EXECUTABLE><IN_FLAG><seq_file><OUT_FLAG><outname><PARAM>
*
*Conventions:
*<FLAG_NAME> <TYPE>       <VALUE>
*<VALUE>:    no_name      <=> Replaced with a space
*<VALUE>:    &nbsp  <=> Replaced with a space
*
*******************************************************
*                  EXECUTABLE                         *
*******************************************************
*name of the executable
*passed to the shell: executable
*
EXECUTABLE   tc_generic_method.pl
*
*******************************************************
*                  ALN_MODE                           *
*******************************************************
*pairwise   ->all Vs all (no self )[(n2-n)/2aln]
*m_pairwise ->all Vs all (no self)[n^2-n]^2
*s_pairwise ->all Vs all (self): [n^2-n]/2 + n
*multiple   ->All the sequences in one go
*
ALN_MODE            pairwise
*
*******************************************************
*                  OUT_MODE                           *
*******************************************************
* mode for the output:
*External methods:
* aln -> alignmnent File (Fasta or ClustalW Format)
* lib-> Library file (TC_LIB_FORMAT_01)
*Internal Methods:
* fL -> Internal Function returning a Lib (Librairie)
* fA -> Internal Function returning an Alignmnent
*
OUT_MODE            aln
*
*******************************************************
*                  IN_FLAG                             *
*******************************************************
*IN_FLAG
*flag indicating the name of the in coming sequences
*IN_FLAG S no_name ->no flag
*IN_FLAG S &nbsp-in&nbsp -> " -in "
*
```

```
IN_FLAG             -infile=
*
*****************************************************
*                  OUT_FLAG                         *
*****************************************************
*OUT_FLAG
*flag indicating the name of the out-coming data
*same conventions as IN_FLAG
*OUT_FLAG    S no_name ->no flag
*
OUT_FLAG            -outfile=
*
*****************************************************
*                  SEQ_TYPE                         *
*****************************************************
*G: Genomic, S: Sequence, P: PDB, R: Profile
*Examples:
*SEQTYPE     S      sequences against sequences (default)
*SEQTYPE     S_P    sequence against structure
*SEQTYPE     P_P    structure against structure
*SEQTYPE     PS     mix of sequences and structure
*
SEQ_TYPE     S
*
*****************************************************
*                  PARAM                            *
*****************************************************
*Parameters sent to the EXECUTABLE
*If there is more than 1 PARAM line, the lines are
*concatenated
*
PARAM  -method clustalw
PARAM   -OUTORDER=INPUT -NEWTREE=core -align -gapopen=-15
*
*****************************************************
*                  END                              *
*****************************************************
```

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.

The WEIGHT parameter 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:

```
PROMPT: t_coffee sample_seq1.fasta –method slow_pair@WEIGHT@OW2
```

Will overweight by a factor 2 the weight of slow_pair.

```
PROMPT: t_coffee sample_seq1.fasta –method slow_pair@WEIGHT@250
```

Will cause every pair of slow_pair to have a weight equal to 250

Plug-Out: Using T-Coffee as a Plug-In

Just because it enjoys enslaving other methods as plug-ins, 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 incorportae 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 ways, 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.

We currently support:

```
Package            Where From
==========================================================
Marna              www.bio.inf.unijena.de/Software/MARNA/download
----------------------------------------------------------
```

Creating Your Own T-Coffee Libraries

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

Using Pre-Computed Alignments

If the method you wish to use is not supported, or if you simply have the alignments, the simplest thing to do is to generate yourself the pairwise/multiple alignments, in FASTA, ClustalW, msf or Pir format and feed them into t_coffee using the *-in* flag:

```
PROMPT: t_coffee –aln=sample_aln1_1.aln,sample_aln1_2.aln –
outfile=combined_aln.aln
```

Customizing the Weighting Scheme

The previous integration method forces you to use the same weighting scheme for each alignment and the rest of the libraries generated on the fly. This weighting scheme is based on global pairwise sequence identity. If you want to use a more specific weighting scheme with a given method, you should either:

generate your own library (cf next section)

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

```
PROMPT: t_coffee –aln sample_aln1.aln –out_lib=test_lib.tc_lib –
lib_only –weight=sim_pam250mt

PROMPT: t_coffee –aln sample_aln1.aln -lib test_lib.tc_lib –
outfile=outaln

PROMPT: t_coffee –aln=sample_aln1_1.aln,sample_aln1_2.aln -method=
fast_pair,lalign_id_pair -outfile=out_aln
```

Generating Your Own Libraries

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

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

```
PROMPT: t_coffee sample_seq1.fasta –method slow_pair –out_lib
slow_pair_seq1.tc_lib –lib_only

PROMPT: t_coffee sample_seq1.fasta –method lalign_id_pair –out_lib
lalign_id_pair_seq1.tc_lib –lib_only
```

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

```
PROMPT: t_coffee sample_seq1.fasta –lib
lalign_id_pair_seq1.tc_lib, slow_pair_seq1.tc_lib
```

Frequently Asked Questions

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

Abnormal Terminations and Wrong Results

Q: The program keeps crashing when I give my sequences

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

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

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

A: 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/) or request T-Coffee to run without using it:

```
PROMPT: t_coffee –pdb=struc1.pdb,struc2.pdb,struc3.pdb -method
sap_pair –cache=no
```

If you do not want to empty your cache, you may also use –cache=update that will only update the files corresponding to your data

```
PROMPT: t_coffee –pdb=struc1.pdb,struc2.pdb,struc3.pdb -method
sap_pair –cache=update
```

Q: The default alignment is not good enough

A: see next question

Q: The alignment contains obvious mistakes

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

Q: The program is crashing

A: If you get the message:

```
FAILED TO ALLOCATE REQUIRED MEMORY
```

See the next question.

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

Q: I am running out of memory

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

```
PROMPT: t_coffee sample_seq1.fasta –special_mode low_memory
```

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

```
PROMPT: t_coffee  sample_seq1.fasta -
method=slow_pair,lalign_id_pair –distance_matrix_mode=idscore -
dp_mode=myers_miller_pair_wise
```

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

Input/Output Control

Q: How many Sequences can t_coffee handle

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

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

Q: Can I prevent the Output of all the warnings?

A: Yes, by setting ***–no_warning***

Q: How many ways to pass parameters to t_coffee?

A: See the section well behaved parameters

Q: How can I change the default output format?

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

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

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

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

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

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

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

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

Q: Is it possible to pipe stuff INTO t_coffee?

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

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

will be equivalent to

```
PROMPT: t_coffee sample_seq1.fasta
```

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

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

For instance:

```
PROMPT: echo -seq=sample_seq1.fasta -method=clustalw_pair |
t_coffee -parameters=stdin
```

Q: Can I read my parameters from a file?

A: See the well behaved parameters section.

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

A: Use the *-run_name* flag.

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

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

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

```
PROMPT: t_coffee Ssample_aln1.aln

PROMPT: t_coffee -infile=Ssample_aln1.aln

PROMPT: 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.

Q: I only want to produce a library

A: use the –lib_only flag

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

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

Q: I want to turn an alignment into a library

A: use the –lib_only flag

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

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

```
PROMPT: t_coffee -aln=sample_aln1.aln -out_lib=sample_lib1.tc_lib
-lib_only -weight=1000
```

Q: I want to concatenate two libraries

A: You cannot concatenate these files on their own. You will have to use t_coffee.

Assume you want to combine tc_lib1.tc_lib and tc_lib2.tc_lib.

```
PROMPT: t_coffee -lib=sample_lib1.tc_lib,sample_lib2.tc_lib –
lib_only -out_lib=sample_lib3.tc_lib
```

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

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

```
PROMPT: t_coffee Ssample_aln1.aln –outfile=outaln
```

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

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

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

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

or

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

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

A: see the next question

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

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

```
PROMPT: t_coffee sample_seq1.fasta -
output=clustalw,html,score_ps,msf
```

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

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

Alignment Computation

Q: Is T-Coffee the best? Why Not Using Muscle, or 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:

```
PROMPT: t_coffee sample_seq1.fasta –method
muscle_msa,probcons_msa, mafft_msa, lalign_id_pair,slow_pair
```

Q: Can t_coffee align Nucleic Acids ???

A: Normally it can, but check in the log that the program recognises the right type (In the INPUT SEQ section, Type: xxxx). If this fails, you will need to manually set the type:

```
PROMPT: t_coffee sample_dnaseq1.fasta –type dna
```

Q: I do not want to compute the alignment.

A: use the -convert flag

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

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

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

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

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

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

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

```
*TC_LIB_FORMAT_02

SeqX resY ResY_index SeqZ ResZ ResZ_index
```

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

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

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

Q: I would like to use structural alignments.

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

Q: I want to build my own libraries.

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

```
PROMPT: 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:

```
PROMPT: 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.

Q: I want to use my own tree

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

```
PROMPT: t_coffee sample_seq1.fasta –usetree=sample_tree.dnd
```

Q: I want to align coding DNA

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

```
PROMPT: t_coffee three_cdna.fasta –method=cdna_fast_pair
```

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

```
PROMPT: t_coffee three_cdna.fasta
```

Note: This method has not yet been fully tested and is only provided "as-is" with no warranty. Any feedback will be much appreciated.

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

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

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

```
PROMPT: t_coffee sample_seq1.fasta –
method=clustalw_pair,clustalw_msa -lib_list=sample_list1.lib_list
-outfile=test
```

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

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

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

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

Using Structures and Profiles

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

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

```
PROMPT: t_coffee sample_seq1.fasta -profile=sample_aln2.aln –
outfile tacos
```

Q: Can I align sequences Two or More Profiles?

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

```
PROMPT: t_coffee -profile=sample_aln1.fasta,sample_aln2.aln –
outfile tacos
```

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

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

```
PROMPT: t_coffee  -profile=sample_profile1.aln,sample_profile2.aln
-special_mode=3dcoffee -outfile=aligne_prf.aln
```

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

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

```
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...ahem...strange., and at least inneficient. Then again I was never struck by the efficiency of the Bush administration.

Q: Can I use a local installation of PDB?

A: Yes, T-Coffe supports three types of installations:

-an add-hoc installation where all your structures are in a directory, under the form pdbid.pdb or pdbid.id.Z or pdbid.pdb.gz. In that case, all you need to do is set the environement variables correctly:

```
setenv (or export)  PDB_DIR="directory containing the pdb
structures"
setenv (or export)  NO_REMOTE_PDB_DIR=1
```

-A standard pdb installation using the all section of pdb. In that case, you must set the variables to:

```
setenv (or export)  PDB_DIR="<some absolute
path>/data/structures/all/pdb/"
setenv (or export)  NO_REMOTE_PDB_DIR=1
```

-A standard pdb installation using the divided section of 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 extract this script from T-Coffee:

```
t_coffee -other_pg unpack_extract_from_pdb
chmod u+x 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. It will 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.

Alignment Evaluation

Q: How good is my alignment?

A: see what is the color index?

Q: What is that color index?

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

```
PROMPT: t_coffee sample_seq1.fasta -output=html
```

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

A book chapter describing the CORE index is available on:

http://www.tcoffee.org/Publications/Pdf/core.pp.pdf

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

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

```
PROMPT: t_coffee -infile=sample_aln1.aln -lib=sample_aln1.tc_lib -
special_mode=evaluate
```

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

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

Q: Can I Compare Two Alignments?

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

```
PROMPT: t_coffee -infile=sample_aln1_1.aln -aln=sample_aln1_2.aln
-special_mode=evaluate
```

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

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

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

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

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

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

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

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

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

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

Improving Your Alignment

Q: How Can I Edit my Alignment Manually?

A: Use jalview, a Java online MSA editor: www.jalview.org

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.

Addresses and Contacts

Contributors

T-coffee is developed, maintained, monitored, used and debugged by a dedicated team that include:

Cédric Notredame

Fabrice Armougom

Des Higgins

Sebastien Moretti

Orla O'Sullivan

Eamon O'Toole

Olivier Poirot

Karsten Suhre

Vladimir Keduas

Iain Wallace

Addresses

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

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.

```
PROMPT: t_coffee –update
```

References

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

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

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

Other useful publications include:

T-Coffee

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

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

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

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

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

Notredame C.

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

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

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

Notredame C. Related Articles, Links

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

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

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

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

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

Mocca

Notredame C. Related Articles, Links

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

CORE

http://www.tcoffee.org/Publications/Pdf/core.pp.pdf

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.

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:

-TM-align 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.

An enormous thanks to these people who believe in free open source code..

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.