

# CoMap Manual

---

Julien Dutheil  
[julien.dutheil@univ-montp2.fr](mailto:julien.dutheil@univ-montp2.fr)

---



# Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>1</b>
<b>2</b>	<b>Data loading .....</b>	<b>3</b>
<b>3</b>	<b>Model specification .....</b>	<b>4</b>
<b>4</b>	<b>Numerical parameters estimation.....</b>	<b>5</b>
<b>5</b>	<b>Substitution mapping.....</b>	<b>6</b>
5.1	Optional commands .....	6
<b>6</b>	<b>Pairwise analysis .....</b>	<b>7</b>
6.1	Performing inter-gene comparisons.....	7
6.2	Optional commands .....	7
<b>7</b>	<b>Clustering analysis .....</b>	<b>9</b>
<b>8</b>	<b>P-value computation .....</b>	<b>10</b>

# 1 Introduction

CoMap performs two kinds of tasks:

*(Weighted) Probabilistic substitution mapping*

Compute all (weighted) number of substitutions occurring on each branch of a tree, for each site of an alignment.

*Co-evolution analysis*

Using the substitution mapping, look for significantly groups of sites departing the null hypothesis of independence. Two kind of analyses are provided: a pairwise analysis, presented in Dutheil et al. (2005), and a clustering analysis in Dutheil and Galtier (2006). In both cases, a parametric bootstrap approach is used to evaluate the significance of groups. Simulation results are written to separate files, a statistics software like R is required to look for the significance. For the clustering analysis, we provide R script to perform these computations. No preliminary knowledge of the R language is required, although it is recommended.

CoMap is a command line program, written in C++ using the Bio++ libraries. It uses the Bio++ syntaxe, so that arguments may be passed as parameter=value options, either directly to the command line or using an option file:

```
comap parameter1=value1 parameter2=value2 ... parameterN=valueN
```

or

```
comap param=option_file.
```

Option files are a list of parameter=value lines, with only one parameter per line. Extra-space may however be included between parameter name, equal sign and value:

```
first_parameter    = value1
second_parameter   = value2
```

Comment may also be included, in either script format

```
# This is a comment
```

or C format

```
/* This is a comment
*/
```

or C++ format

```
// This is a comment
```

Command line and file options may be combined:

```
comap param=option_file parameterX=valueX
```

In case of parameterX is specified in both option file and command line, the command line value will be used. This allows to run CoMap several time by changing a single option, like the kind of mapping for instance.

The next chapters describes the whole set of options available in CoMap. The type of parameter value expected is defined as:

- [chars]     A character chain
- [path]     A file path, may be absolute or related to the current directory
- [int]       An integer
- [int], [int>0], [int>=0], [int[2,10]]  
            An integer, a positive integer, a positive non-null integer, an integer falling between 2 and 10
- [real], [real>0], etc  
            A real number, a positive real number, etc.
- [boolean]  
            A boolean value, may be one of 'yes', 'no', 'true' or 'false'
- [xxx|xxx|xxx]  
            A set of allowed values
- list[type]  
            A list of values of specified type, separated by commas.

From version 1.2, CoMap can use Bio++ macro system. This is possible to recall anywhere the value of an option by using \$(parameter).

```
model=JTT92
rat_distribution=gamma
output.tree = mydata$(model)$(rate_distribution).dnd
```

You can use this system to define global variables:

```
data=LSU
sequence.file=$(data).fasta
input.tree=$(data).dnd
output.infos=$(data).infos
```

## 2 Data loading

`alphabet = [DNA|RNA|Protein]`

The alphabet to use when reading sequence.

`sequence.file=[path]`

The sequence file to use (sequences must be aligned!)

`sequence.format = [Mase|Fasta|Phylip|Clustal|DCSE]`

The alignment file format.

`sequence.format_mase.site_selection = [chars]`

Meaningful only for mase format. Specify a name for a site set to use.

`sequence.format_phylip.order = [interleaved|sequential]`

Meaningful only for phylip format. Tells if sequences are interleaved or sequential format.

`sequence.format_phylip.ext = [classic|extended]`

Meaningful only for phylip format. The 'classic' option corresponds to old phylip format, with names up to ten characters. The 'extended' option corresponds to the behaviour of PAML, which allows sequence name of any size, separated from the sequence with at least two spaces.

`sequence.sites_to_use = [all|nogap|complete]`

Tells which sites to use. The 'nogap' option removes all sites containing at least one gap, and the 'complete' option removes all sites containing at least one gap or one generic character, as 'X' for instance.

`sequence.max_gap_allowed=101%`

Only works when the 'all' option is selected. specify the maximum amount of gap allowed by site, given as a number of sequence or a percentage. Sites not succeeding the criterion will not be included in the analysis.

`tree.file = [path]`

The phylogenetic tree file to use. Branch lengths are optional. Only newick format is supported for now.

### 3 Model specification

```
model = [JCnuc|K80|T92|HKY85|F84|TN93|GTR|JCprot|DS078|JTT92|empirical]
```

Specify the substitution model to use. For proteins, the DCMutt method is used for JTT92 and DSO78. You can use the 'empirical' option to specify another model, as a text file in PAML format.

```
kappa=[real>0]
```

Initial value or fixed value for parameter kappa (transition/transversion ratio) in models K80, T92, HKY85 and F84

```
kappa1=[real>0]
```

Initial value or fixed value for parameter kappa1 in model TN93

```
kappa2=[real>0]
```

Initial value or fixed value for parameter kappa2 in model TN93

```
theta=[real[0,1]]
```

Initial value or fixed value for parameter theta (GC content) in model T92

```
piA,piC,piG,piT = [real[0,1]]
```

Equilibrium frequencies of nucleotides A, C, G and T/U in models HKY85, F84, TN93 and GTR

```
a,b,c,d,e = [real>0]
```

Parameters of the GTR model

```
model.use_observed_freq = [boolean]
```

Tell if we have to use observed frequencies as parameter values (for piA, piT, piG, piC and theta). For proteins, this is equivalent to the -F model family.

```
model_empirical.file = [path]
```

Meaningful only if the 'empirical' option is selected. Specify the path where to find the model (in PAML format).

```
rate_distribution = [constant|gamma]
```

Specify the rate across sites distribution, as constant or gamma.

```
rate_distribution_gamma.alpha = [real>0]
```

The gamma distribution's shape parameter

```
rate_distribution.classes_number = [int>=2]
```

The number of classes to use in discretization.

## 4 Numerical parameters estimation

CoMap can (re-)estimate numerical parameters for you before any analysis. These parameters include

- Branch lengths
- Entries of the substitution matrices, excepted base frequencies (piA, piC, piG and piT parameters, which must be taken as there observed values)
- Parameters of the rate distribution (currently only the shape parameter of the gamma law)

`optimization = [boolean]`

Tells if numerical parameters should be estimated. Topology estimation is not supported.

`optimization.method=[NB|NND]`

The optimization method to use. NB (default) stands for Newton-Raphson for branch lengths + Brent for other parameters, and the NND options uses Newton-Raphson for all parameters, with numerical derivatives for non-branch lengths parameters.

`optimization_NB.nstep=[int>=1]`

Number of progressive steps to perform during optimization. If nstep=3 and precision=E-6, a first optimization with precision = E-2, will be performed, then a round with precision set to E-4 and finally precision will be set to E-6. This approach generally saves significant computation time.

`optimization.profiler = [[path]|std]|none`

A file where to dump optimization steps (a file path or std for standard output or none for no output).

`optimization.message_handler = [[path]|std]|none`

A file where to dump warning messages.

`optimization.max_number_f_eval = [int<0]`

The maximum number of likelihood evaluation to perform.

`optimization.ignore_parameter = list[float[0,1]`

A list of parameters to ignore during the estimation process.

`optimization.tolerance = [float>0]`

The precision on the log-likelihood to reach.

`output.tree.file = [[path]|none]`

File path where to write the optimized tree.

`output.infos = [[path]|none]`

A text file containing several statistics for each site in the alignment. These statistics include posterior rate, rate class with maximum posterior probability and whether the site is conserved or not.



## 5 Substitution mapping

Options in this chapter are for substitution vectors computation.

`input.vectors.file = [[path] | none]`

Restart an analysis by specifying the already computed vectors. Otherwise, compute vectors using the following options.

`nijt = [laplace | simple | aadist]`

The kind of mapping to perform. ‘laplace’ option perform exact mapping, as in Dutheil et al. (2005). ‘simple’ performs a naive mapping, as in Tuffry and Darlu (2000). ‘aadist’ performs a weighted mapping, as in Dutheil and Galtier (2007). Currently available only for proteins.

`nijt_aadist.type =`

`[grantham | miyata | grantham.volume | grantham.polarity | charge | klein.charge | user1 | user2]`

Specify the type of weight to use. The ‘user1’ option is for computing a simple distance from a AAIndex1 file, ‘user2’ uses a AAIndex2 distance file.

`nijt_aadist.type_user1.file = [path]`

File path toward a AAIndex1 file. Used only if ‘nijt\_aadist.type=user1’.

`nijt_aadist.type_user2.file = [path]`

File path toward a AAIndex2 file. Used only if ‘nijt\_aadist.type=user2’.

`nijt_aadist.sym = [boolean]`

Tell if symetric matrices must be used. This option should be set to "no" for testing compensation.

`output.vectors.file = [[path] | none]`

Where to write the substitution mapping.

`output.tags.file = [[path] | none]`

A tree file in newick format, with node ids instead of bootstrap values. The node ids are the same as in the the vector file.

`output.tags.translation = [[path] | none]`

Write a file whith the correspondance between leaf names and ids.

### 5.1 Optional commands

`nijt_laplace.trunc = [int>0]`

Where to trunc the series when estimating exact number of substitutions. A value of 10 should be fine.

`nijt.average = [boolean]`

Tell if mapping should be averaged over all ancestral states (probabilistic mapping). Otherwise use ancestral states reconstruction (naive mapping). In most case, you should leave the default value (yes). NB: only marginal ancestral state reconstruction is implemented.

`nijt.joint = [boolean]`

Tell if joint probabilities are to be use, otherwise us emarginal probabilities. This option is for method comparisons, a ‘yes’ value is suitable in most cases.

## 6 Pairwise analysis

`statistic = [correlation|none]`

If 'none' is selected, no pairwise analysis is performed.

`statistic.output.file = [path]`

Where to write the statistic value for each pair of sites.

`statistic.null = [boolean]`

Tell is the null distribution of the statistic must be computed, using parametric bootstrap. The number of simulations performed is the product of two numbers, adjusting the amount of CPU and RAM to use.

`statistic.null.nb_rep_CPU = [int>0]`

Increase this parameter to take less memory (slow the program)

`statistic.null.nb_rep_RAM = [int>0]`

Increase this parameter to speed the program (need more memory)

`statistic.null.output_file = [path]`

Where to write the null distribution

### 6.1 Performing inter-gene comparisons

It is possible to compare all sites from one data set with all sites from a second data set (inter-gene analysis).

`sequence.file2 = [[path]|none]`

The path toward the second file. All previous options can be set up for second file, just append '2' at option names. The default is to use options of file1 for file2.

### 6.2 Optional commands

`statistic.min = [float]`

Write only pairs with a statistic greater or equal to this value.

`statistic.min_rate = [float>0]`

Write only pairs with a posterior rate greater or equal to this value.

`statistic.min_rate_class = [int>0]`

Write only pairs with a minimum rate class greater or equal to this value (first class is 0).

`statistic.max_rate_diff = [float]`

Write only pairs with rates that do not differ more than this value (-1 -> write all pairs).

`statistic.max_rate_class_diff = [int]`

Write only pairs with rate classes that do not differ more than this value (-1 -> write all pairs).

`statistic.null.cumul = [bool]`

[deprecated] Tell if an histogram of the distribution should be returned instead of printing all simulated pairs.

`statistic.null.lower = [float]`

[deprecated] Lower bound of histogram.

`statistic.null.upper = [float]`

[deprecated] Upper bound of histogram.

```
statistic.null.nb_int = [int<0]  
    [deprecated] Number of intervals in histogram.
```

## 7 Clustering analysis

`clustering.distance = [cor|euclidian|comp|none]`

Distance to use: cor (correlation), euclidian, comp (compensation) or none (no clustering).

`clustering.scale = [boolean]`

Tell is mapping should be normalized (each row (=branch) will be divided by its sum).

`clustering.method = [complete|sum]`

Clustering algorithm: complete linkage or sum. The sum option should be used only when testing compensation, i.e. in conjunction with the `'nigt_aadist.sym=no'` and `'clustering.distance=comp'` options.

`clustering.output.matrix.file = [[path]|none]`

Where to write the distance matrix (in phylip format).

`clustering.output.tree.file = [[path]|none]`

Where to write the clustering tree (newick format).

`clustering.output.groups.file = [[path]|none]`

Where to write the clusters (CSV format).

`clustering.null = [boolean]`

Tell if the null distribution of clusters must be computed.

`clustering.null.number = [int>0]`

Number of data sets to simulate.

`clustering.null.output.file = [path]`

Where to write the simulated clusters (CSV format).

## 8 P-value computation

CoMap do not compute p-values from the simulation. This goal is achieved by two R scripts, distributed along with the program. The script ‘CoMapFunctions.R’ contains several functions performing the computation. The script ‘computePValues.R’ is the one to launch. It calls the previous one, so they must be in the same directory.

Edit the first section of the ‘computePValues.R’ so that it matches your files, and then run it with the command

```
R --vanilla < computePValues.R
```