CoMap Manual

Julien Dutheil
julien.dutheil@univ-montp2.fr

1

This is the manual of CoMap, version 1.3.0, 23 January 2008.

Table of Contents

1 Introduction

CoMap performs two kinds of tasks:

(Weighted) Probabilistic substitution mapping
: Compute all (weighted) number of subsitutions occuring 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 independance. Two kind of analyses are provided: a pairwise analysis, presented in Dutheil et al. (2005), and a clustering analysis in Dutheil ang Galtier (2007). 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 significancy. For the clustering analysis, we provide R script to perform theses computations. No preliminary knowledge of the R language is required, although it is recommanded.

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 integet, an integer folling 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 comas.

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

```
model.name=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 interleave 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 wich 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

3.1 Substitution model

`model.name = [JCnuc|K80|T92|HKY85|F84|TN93|GTR|JCprot|DSO78|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.

`model.kappa=[real>0]`
: Initial value or fixed value for parameter kappa (transition/transversion ratio) in models K80, T92, HKY85 and F84

`model.kappa1=[real>0]`
: Initial value or fixed value for parameter kappa1 in model TN93

`model.kappa2=[real>0]`
: Initial value or fixed value for parameter kappa2 in model TN93

`model.theta=[real]0,1[]`
: Initial value or fixed value for parameter theta (GC content) in model T92, HKY85, F84, TN93 and GTR.

`model.theta1 = [real]0,1[]`
: Proportion of G / (G + C) in models HKY85, F84, TN93 and GTR

`model.theta2 = [real]0,1[]`
: Proportion of A / (A + T/U) in models HKY85, F84, TN93 and GTR

`model.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 '`theta`', '`theta1`', '`theta2`', etc). 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).

3.2 Non-homogeneous models

You can specify a wide range of non-homogeneous models. The novelty from the homogeenous case is that the likelihood depends on the position of the root. Root frequencies are hence distinct parameters:

`nonhomogeneous.root_freq=[balanced|observed|init|balancedGC|observedGC|initGC]`
: The '`balanced`' option set all the frequencies to 1/size of the alphabet. The '`balancedGC`' option is for nucleotides only, and set the G content equal to the C content equal to 0.5. The '`observed`' option set frequencies equal to their values in the data set,, and '`observedGC`' does the same for the GC content. The '`init`' and '`initGC`' options allows you to set the values of the frequencies by hand.

`model.ancA, ancC,..., ancA, ancR, ancN, ..., ancTheta`
: Initial values of ancestral frequencies if the '`init`' or '`initGC`' have been specified.

`nonhomogeneous = [no|one_per_branch|general]`
: Homogeneous model, Galtier & Gouy 1997 model family or the general case.

3.2.1 One-per-branch non-homogeneous models

This option share the same parameters as the homogeneous case, since the same kind of model is used for each branch. The additional options are the following:

`nonhomogeneous_one_per_branch.shared_parameters = [list]`
: List the names of the parameters that are shared by all branches. In Galtier & Gouy model, that would be model.kappa, since only the theta parameter is branch-specific.

3.2.2 General non-homogeneous models

Bio++ provides a general syntaxe to specify almost any non-homogeneous model.

`nonhomogeneous.number_of_models`
: Set the number of distinct models to use.

You now have to configure each model individually, using the syntaxe introduced for the homogeneous case, excepted that model will be numbered, for instance:

```
model1.name = T92
model1.theta = 0.39
model1.kappa = 2.79
```

The additional option is available to attach the model to branches in the tree, specified by the id of the upper node in the tree:

`model1.nodes_id = 1,5,10:15,19`
: Specify the ids of the nodes to which the node is attached. Id ranges can be specified using the '`begin:end`' syntaxe.

You can also make a given model share parameters with another one by writting for instance:

```
model2.name = T92
model2.theta = 0.39
model2.kappa = model1.kappa
```

Finally, you may find useful the following options:

`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.

3.3 Rate across site distribution

`rate_distribution = [constant|gamma{+invariant}]`
: Specify the rate across sites distribution, as constant or gamma, optionally with an invariant class.

`rate_distribution.alpha = [real>0]`
: The gamma distribution's shape parameter

`rate_distribution.classes_number = [int>=2]`
: The number of classes to use in discretization (not including the invariant class if any)

`rate_distribution.p = [real[0,1]]`
: Proportion of invariant sites

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, included base frequencies. values)
- Parameters of the rate distribution (shape parameter of the gamma law, proportion of invariant sites).

`optimization = [boolean]`
: Tells if numerical parameters should be estimated. Topology estimation is not supported.

`optimization.method=[DB|fullD]`
: The optimization method to use. DB (default) uses derivatives-based optimization for branch lengths + Brent for other parameters, and the fullD options uses derivatives for all parameters, with numerical derivatives for non-branch lengths parameters.

`optimization_DB.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 approcah generally saves significant computation time.

`optimization.method.derivatives = [newton|gradient]`
: Derivatives-based algorithm to use. The Newton algorithm uses derivatives up to the second order, and is the recommanded choice. The gradient algortihm uses only first order derivatives and can also be used.

`optimization.final = [powell|simplex]`
: Optinal final optimization step, useful if numerical derivatives are to be used. Leave the field empty in order to skip this step.

`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.

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

`analysis = pairwise`
: Use this option for performing a pairwise analysis. If another option or '`none`' is selected, no pairwise analysis is performed.

`statistic = [correlation|compensation|cosubstitutions]`
: The coevolution statistic to use. The '`correlation`' option is to be used in order to perform the MBE 2005 analysis. Uses option '`compensation`' to perform a pairwise analysis with the compensation statistics introduced in the BMC Evol Biol 2007 paper. You can also use the '`cosubstitution`' option to perform Tuffery & Darlu's MBE 2000 analysis.

`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

`analysis = clustering`
: Use this option for performing a clustering. If another option or '`none`' is selected, no pairwise analysis is performed.

`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]`
: Clustering algorithm: complete linkage. Other linkage types are available, but the complete one gives the better results.

`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 Candidate groups analysis

This method is under development and is not yet available, sorry.

9 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 is matches your files, and then run it with the command

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