CoMap Manual

Julien Dutheil
dutheil@evolbio.mpg.de

1

This is the manual of CoMap, version 1.5.5.

Table of Contents

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 analyzes are provided: a pairwise analysis, presented in Dutheil et al. (2005), and a clustering analysis in Dutheil and 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 significance. For the clustering analysis, we provide R script to perform theses 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++ syntax, so that arguments may be passed as parameter=value options, either directly to the command line or using an option file. For more information about this very general syntax, please read the Bio++ Program Suite manual, available online at `http://home.gna.org/bppsuite`. You can also download a PDF version here `http://home.gna.org/bppsuite`. Since it uses the Bio++ libraries, CoMap shares several options with the BppML program, to read sequences and specify models for instance. We will only report the difference between CoMap and BppML in this manual.

2 Data loading

CoMap inputs two kinds of data: a sequence alignment file and a phylogenetic tree. The corresponding options are fully described in the Bio++ Program Suite manual.

CoMap adds an option to remove conserved sites from the analysis. The removal will occur after parameter optimization, if relevent.

3 Model specification

CoMap support all substitution models available in Bio++, including non-homogeneous models. Options for specifying a model are described in full in the Bio++ Program Suite manual.

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).

The corresponding options are described in the Bio++ Program Suite manual. Note however that CoMap do not perform topology estimation. The tree topology will is considered to be know prior to the analysis. The tree can however contain multifurcations (WARNING: not fully tested, please report any bug you may found!).

5 Substitution mapping

Options described in this part are specific to CoMap. Options in this chapter are for the computation of substitution vectors.

`input.vectors.file = {{path}|none}`
: Restart an analysis by specifying the already computed vectors. Otherwise, compute vectors using the following options.

`output.vectors.file = {{path}|none}`
: Where to write the substitution mapping.

`nijt = {Naive|Laplace|Uniformization|Decomposition}`
: The kind of mapping to perform. `Laplace(trunc={int>1})` option perform exact mapping, as in Dutheil et al. (2005). The `trunc` option set where to trunc the series (default to 10). Note that this substitution count is rather slow, the uniformization or decomposition methods are faster and more accurate. This option is only there for legacy purpose. `Naive(weight=...)` performs a naive mapping, as in Tuffery and Darlu (2000). `Uniformization(weight=...)` [default] performs exact mapping, with the uniformization method (See Tataru and Hobolth, 2011). `Decomposition(weight=...)` performs exact mapping, with the decomposition method (See Tataru and Hobolth, 2011). Currently available only for proteins.

The Naive, Uniformization and Decomposition methods all allow to perform weighted substitution mapping, by taking a `weight` argument. Only two options are available for now:

`None` The default, no weighting is used, all substitution weight the same.

`AAdist` Amino-acid distance. This option takes further arguments:

`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.

`file = {path}`
: File path toward a AAIndex1 and AAIndex2 file, ignored otherwise.

`sym = {boolean}`
: Tell if symmetric matrices must be used. This option should be set to "no" for testing compensation.

5.1 Optional commands

`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 = {bool}`
: Tell if joint probabilities are to be use, otherwise use marginal probabilities. This option is for method comparisons, a 'yes' value is suitable in most cases.

5.2 Site specific statistics

Some of the calculations made by CoMap on each site can be output to a file. This file can be used for a posteriori analysis or check of results.

`output.infos = {path}`
: Path of the output file.

The output file is tab-separated spreadsheet, with one line per site. Site names are identical to the ones output in the coevolution analysis. Columns are: IsComplete (yes if the site has no missing data), IsConstant (yes if the site is monomorphic), RC (the rate class, defined as the category with maximal posterior probability), PR (the posterior estimate of the site-specific rate, averaged over all rate classes), N (the norm of the (weighted) substitution vector, logLn (the log-likelihood for the site).

5.3 Ancestral sequences reconstruction

CoMap can also perform ancestral sequence reconstruction. Ancestral sequences are not used in the analysis, but can be useful for interpreting results. Marginal reconstruction is performed, for more advanced techniques and options, consider the bppAncestor program from the Bio++ Program Suite. Note that ancestral states are output for analyzed positions only.

`asr.method = {none|marginal}`
: Method to use, only marginal reconstruction is available for now.

`output.sequence.file = {path}`
: Where to write reconstructed sequences, together with extant sequences.

`output.sequence.format = {string}`
: All Bio++ format are supported.

5.4 Output node ids

Nodes and branches are automatically labeled by CoMap, and the resulting ids are used to report substitution counts and ancestral sequences. These ids can be output in the form of a tree file, with nodes at leaves as leaf names and inner nodes as "bootstrap" values.

`output.tags.file = {path}`
: Where to write the annotated tree, in newick format.

`output.tags.translation = {string}`
: Write a text file displaying the leaf ids and the corresponding orginal sequence names.

These files are required for visualization of candidate groups (see the example/Visualization folder in source distribution).

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

`statistic.null.compute_pvalue = {bool} (statistic.null=yes)`
: Shall we use the null distribution to compute p-values? From version 1.4.0, CoMap can now compute built-in p-values, without relying on an external R-script.

`statistic.null.nb_rate_classes = {int} (statistic.null.compute_pvalue = yes)`
: Specifies how many classes should be used when conditioning of pairs rate. A value of 0 or 1 implies no conditionning.

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 = {boolean}`
: [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), euclidean, comp (compensation) or none (no clustering).

`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 used to test candidate groups of sites. It perform a more powerful test by taking into account the precise rate of each site, and no longer relies on the minimum rate of the group as in the pairwise or clustering method. As a consequence, it is much slower, and should therefore only be used when some candidate groups are to be tested. It can't be used in an exhaustive manner.

Important note: This method does not require the R scripts to be used! It directly outputs p-values :)

`candidates.input.file = {path}`
: The file to read. It must be in CSV-like format, with one group per line, and column names.

`candidates.input.column_name = {string}`
: The name of the column that will contain groups. Groups should be described as [pos1;pos2;etc], refering to alignment positions, starting at 1.

`candidates.input.column_sep = {char}`
: The character used to separate columns in the input file.

`candidates.omega = {float}`
: The width of the window for telling whether two sites have similar rates or not. A value of 0.2 is usually good enough. The smaller the value, the more precise the computations, but the slower the execution time.

`candidates.null.min = {int>0}`
: Minimum number of simulations to perform for each group.

`candidates.null.nb_rep_RAM = {int>0}`
: Controles the number of sites to simulate at a time. The higher the value, the faster the execution, but at the expense of more memory usage.

`candidates.null.verbose = {int>=0}`
: Controles the amount of output to terminal.

`candidates.output.file = {path}`
: The file to write, identical to the input one but with extra columns, in CSV format.

`candidates.output.column_sep = {char}`
: The column separator for output file.

9 Output format and P-value computation

CoMap computes built-in p-values for the pairwise and candidate group analyses. For group detections, p-value computation is performed by two R scripts, distributed along with the program. The reason for this is that there are different ways to compute p-values, depending on the null-hypothesis to test. Getting the simulation result being the most expensive in terms of computer resources, I preferred to store them in files, and then play with R to get the p-values. It also allows to visualize the null-distribution and get more "feeling" about the data.

9.1 Output format

With the pairwise and clustering analysis, CoMap outputs two files, one for the real data, and one for simulations. The two files are in the same format, a simple text file with columns separated by tabs. Each row stands for one group. The available columns are:

`Stat` Contains the value of the statistic for the group.

`Nmin` Contains the minimum norm of the group. The norm of the substitution vector is a general evolutionary rate measure (See Dutheil 2008, JME for more information).

The output file for real data contains an additional column, `Group`, which describes the corresponding sites in the alignment. It is a text string, beginning with "[" and ending with "]", with site positions separated by semi-colons. Site numbering begins with 1, and is according to the original alignment, accounting for any gap or ignored positions in the analysis.

For the clustering analysis, the column `Size` also contains the size of the group, that is the number of sites, and the column `Dmax` contains the height of the supporting node in the clustering tree. The simulation output for clustering also contains an additional column named `Rep` that contains the index of the simulated data set.

Additional columns, not used to compute p-values:

`RCmin` Contains the minimum rate class for the group. the number of classes depends on the option specified as input.

`PRmin` Similar to `RCmin`, but with the posterior rate instead of the rate class.

`Const` Tell if the group contains at least a totally conserved (=constant) site.

9.2 Available R scripts

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
```

10 Mica: Mutual Information Coevolution Analysis

The Mica program performs several standard mutual information (MI) analyses. Basically, it computes the MI value for all pairs of sites in an alignment, and optionally compute the MI value on randomized data. The resulting values can be compared in R in order to get a simple p-value. Mica also computes various rate measures, so that the CoMap R scripts and algorithms can be used with Mica's output. Mica can generate replicates using either parametric bootstrap, like CoMap, or non-parametric bootstrap like in Caporaso et al (2008). It can also compute corrected MI (MIp from Gloor et al), and the associated p-values using a Z-score approach.

10.1 Input data.

Mica needs a sequence alignment, which can be read as in CoMap, with the same formats available. It works with all types of alphabets. Optionally, Mica can also take a phylogenetic tree and a substitution model. Here again, all models supported in CoMap are supported by Mica. If such model and tree are provided, Mica will perform substitution mapping and compute the norm of the substitution vector for each site, a better measure of evolutionary rate than the Shannon entropy. Such a model is also required in order to perform parametric bootstrap. The argument `use_model = yes` (no by default) enables the use of a substitution model (Jukes-Cantor by default).

10.2 Output results.

Mica outputs a CSV file with all tested pairs, with various statistics, including:

`MI` The mutal information for the site.

`APC` The average product correction of Dunn et al (2008). MIp can be computed using the formula $MI - APC$.

`RCW` The row-column weighting, of Gouveia-Oliveira (2007). The corrected MI is computed using the formula MI/RCW.

`Hjoint` The joint entroy of the two sites.

`Hmin` The minimum entropy of the two sites, $Hmin = \min(H_i, H_j)$.

If a model and a phylogenetic tree were set up, the following extra information will also be available:

`Nmin` The minimum norm of substitution vector for the pair.

10.3 Assessing significance of the coevolution statistics.

Mica implements four possible ways to compute a significance value (p-value), which is specified by the command `null.method`. This command accepts 5 options:

`none` Will disable the computation of the null distribution.

`permutations`
: Use simple permutations to compute p-values. This method does not account for shared ancestry and should only be used for didactic reasons. This option will add two colums, namely `Perm.p.value` and `Perm.nb`, with similar meaning as the bootstrap columns.

`nonparametric-bootstrap`
: Will sample sites from the original data set.

`parametric-bootstrap`
: Performs simulations using the tree and model set as input.

`z-score` Use the set of all pairs in the original data set as a null distribution.

When a bootstrap analysis is performed, the following options are available:

`null.nb_rep_CPU = {int > 0}, null.nb_rep_RAM = {int > 0}`
: will perform CPU * RAM replicates in total. Increasing the RAM components will be faster but require more memory, while increasing CPU will demand more CPU time, but will use less memory.

`null.output.file = {path}`
: The file where all samples from the null distribution should be written. Setting `none` will disable output. The output file can be used for making the p-value computation by hand, using the R software for instance.

`null.compute_pvalues = {bool}`
: Tells if p-values should be computed from the null distribution. If set to yes, then two columns will be added to the output file: `Bs.p.value`, which stores the actual computed p-value, and `Bs.nb`, storing the number of replicates on which the computation is based. This number of replicates is the specified one when no rate comditioning is performed (see below). Otherwise it depends on the number of classes used. P-values are always computed when Z-scores are used.

The bootstrap analyses will only be used for assessing the significance of MI. A Z-score approach is to be used for computing the p-value associated to the corrected MIs. One therefore specifies what is the statistic for wich p-value should be computed:

`null.method_zscore.stat = {MI|MIp|MIc}`
: The statistic for which p-values should be computed.

10.3.1 Rate conditioning

For all approaches but permutations, more specific null distributions can be obtained by conditioning over the evolutionary rate:

`null.nb_rate_classes = {int > 0}`
: If greater than 1, enables rate conditioning when computing the p-value. The classes are computed by cutting the interval [0, max observation] into the specified number of categories. By default, the minimum entropy is used as a rate measure. If a model is specified, the the minimum norm is used instead.

Additional option for the permutation analysis:

`max_number_of_permutations = {int >= 0 }`
: Set the number of permutations that can be performed. The minimal p-value that will be obtained is obtained by $1/(n+1)$. For speeding up computations, the p-value of the pairs that are far from the tail of the distribution (and therefore most-likely non-significant) will be computed from less observations (the computation will stop when at least 5 points from the null distribution with a higher statistic than the observed one are found).