LMAP: Lightweight Multigene Analyses in PAML

MANUAL

Version: 1.0.0, 3rd June, 2016

Authors:

Emanuel Maldonado[1], Daniela Almeida[1,2], Tibisay Escalona[1,2], Imran Khan[1,2], Vitor Vasconcelos[1,2] and Agostinho Antunes[1,2].

1 CIIMAR/CIMAR – Interdisciplinary Centre of Marine and Environmental Research, University of Porto, Porto, Portugal
2 Department of Biology, Faculty of Sciences, University of Porto, Porto, Portugal

Table of Contents

Figures and Tables

List of Notes

1. INSTALLATION

1.1. LMAP Archive

The Table 1, presents the applications (8) and modules (4) included in the LMAP archive (LMAPvx.x.x.zip).

Table 1. *Summary of LMAP applications.*

LMAP Application	Functionality	LMAP Module
gmap.pl	Creation of the directory structure and phylogenetic tree labeling.	MyPAMLInfo.pm, MyPhylo.pm, MyUtil.pm
cmap.pl	Change *codeml* control files parameters.	MyUtil.pm
mmap.pl	Execute and monitor *codeml* instances.	MyPAMLInfo.pm, MyNotify.pm, MyUtil.pm
imap.pl	Extract maximum likelihood (ML) parameter estimates.	MyPAMLInfo.pm, MyUtil.pm
omap.pl	Organize and estimate LRTs.	MyUtil.pm
lmap.pl	All above (gmap.pl+cmap.pl+mmap.pl +imap.pl+omap.pl)	MyUtil.pm, MyNotify.pm, MyPAMLInfo.pm
install.pl	Install LMAP requirements.	-
configure.pl	LMAP configuration.	MyNotify.pm, MyUtil.pm

Beyond these central elements, the archive includes a folder containing a dataset (“ExampleDataset”) for which results were calculated (“ExampleDatasetResults”) with LMAP applications. Additionally, the command which produced these results is indicated in the file named “lmap.command.txt” and explained in the accompanying “README.txt” file.

The dataset included demonstrates the usefulness and simplicity that LMAP applications provide. Its sole purpose is to help users to understand how to prepare input files and to show how LMAP works by enabling immediate trial and testing.

1.1.1. Requirements

LMAP applications are written in Perl and it is assumed that Perl is already installed on the user workstation. CPAN (https://metacpan.org/) should also be already configured in your workstation. LMAP requires the following external modules and programs.

***Table 2**. Required PERL CPAN Modules.*

Module	Observations
IO::All	Input/Output
Email::Sender	Email notifications
Email::MIME	Email notifications
Sys::Info	System information
Term::ReadKey	Terminal operations
Thread::Semaphore	Threads
Statistics::Distributions	LRT estimations
Math::Cephes	LRT estimations
Bio::TreeIO	Phylogeny editing operations (BioPerl [1])
File::Copy	File operations
File::Copy::Recursive	File operations

***Table 3**. Required software.*

Software	Information
PAML package (min. v.4.6) [2]	http://abacus.gene.ucl.ac.uk/software/paml.html
screen	https://www.gnu.org/software/screen/
sendmail(*)	http://www.sendmail.com/sm/open_source/

(*) – Assuming the user will benefit from email notification, this is listed as required software and is installed automatically (see section **1.2.1.1**). If otherwise the email notification is not required, the installation of this utility can be discarded (see section **1.2.2** step 2).

1.2. Instructions

These instructions make use of the Ubuntu package manager: APT (*apt-get* command).
If you are using a different Linux distribution, you will need to install programs manually or with an appropriate package manager available to your system.

NOTES FOR MAC OS users:

For MAC OS users, the Xcode Developper tools (https://developer.apple.com/xcode/) are necessary, which will make Perl and other tools available (like *screen* and *sendmail*). See http://learn.perl.org/installing/osx.html.
The required programs (Table 3) may have to be installed manually, if the MacOS package manager is not providing their installation.

In Mac OS systems it may be possible that the install scripts and applications will not execute due to different Perl configurations or multiple Perl installations. In this sense, it may be either required to (i) change the first line occurring in each LMAP application and installation scripts or to (ii) just run each application with the *perl* command.

(i) The first line appearing at the top of each LMAP executable (files ending in ".pl") is

```
#!/usr/bin/perl -w
```

This line should be changed to:

```
#!/usr/bin/env perl -w
```

(ii) Alternatively, all LMAP applications can be executed with *perl* itself:

```
$ perl <appname.pl>
```

See also a list of available package managers:
https://en.wikipedia.org/wiki/List_of_software_package_management_systems

See also how to install BioPerl
http://www.bioperl.org/wiki/Installing_BioPerl_on_Unix

To install LMAP you may either (**1.2.1.**) run the included installation scripts; or (**1.2.2.**) install all the requirements manually.

1.2.1. Install using the LMAP programs

The easiest way to install LMAP package, is to run the *install.pl* (version: 1.0.0 Aug 14th, 2015) and *configure.pl* (version: 1.0.0 Aug 14th, 2015) programs located at the base of the LMAP unzipped directory.

```
$ cd LMAPv1.0.0/LMAP
```

1.2.1.1 Install Requirements

```
$ sudo ./install.pl
```

It will automatically install all the LMAP requirements previously listed (Tables 2-3). If any modules or programs were previously installed, they will not be reinstalled.

NOTE 1: You may need to run the install script twice, if CPAN was not previously configured in your computer/account.

This install procedure will do the following:

1) install CPAN Modules (Table 2)
This is done through specific CPAN oriented modules that allow the installation of other modules.

2) install Programs from repositories (Table 3)

This is done automatically for each program by using the *apt-get* command.

1.2.1.2. Configure LMAP

After the requirements installation is complete, the following step is to configure LMAP. This step can be repeated any number of times, as per the user requirements, to change any previous configuration.

```
$ ./configure.pl
```

or;

```
$ sudo ./configure.pl
```

This application will do the following:

1) ask the user for configurations:

i. Disable use of terminal colors;
ii. Select location of PAML executables;
iii. Email notification settings (optional step; see below, Table 4);
iv. Select final location of LMAP applications

2) generate a configuration file in the user's $HOME directory

3) move all the LMAP applications and modules into a user selected (binaries) location.

To have LMAP available throughout any directory location in your account/workstation, LMAP should be placed in a binaries location. Still, this is not a requirement.

In the first command case, the user will configure LMAP to the user's $HOME directory, usually in $HOME/bin. In the second case with *sudo* command, the configuration of LMAP applications will be allowed in a root directory e.g.: /usr/local/bin/, thus available to any user account in the same workstation.

However, assuming there are multiple user accounts that will use LMAP, each account requires the LMAP configuration file. In this sense, since LMAP was already configured "system-wide" (second case), the application should now be executed in each account without *sudo* command (first case) and the last step (iv) requiring the selection of LMAP applications location, should be ignored. Here, to this end, the user must select the option "Do nothing, I will copy LMAP applications...". After this, the applications are ready and any user will have his own configuration settings in his/her own account. In this situation and in the first case, any user may configure LMAP in his account and not require administrative privileges. These are only required when installing/configuring LMAP in the system directories.

NOTE 2: to configure LMAP in a root directory (e.g.: /usr/local/bin/), run this configuration script with *sudo* command.

To enable email notification, the *sendmail* utility must have been installed (see Table 3 and sections 1.2.1.1 and/or 1.2.2), and the settings in Table 4 are needed before proceeding to this configuration step. These settings are related to the required CPAN modules (Table 2). The configuration of email notification is optional. While executing this application, the user has the possibility to skip this step. If the user does not need email notifications, then

this step is not required as also it is not required the installation of the *sendmail* utility.

Table 4. *Information required to enable email notifications.*

Information	Dummy Examples
SMTP server hostname and the required port number (or by default 25)	smtp.uni.fac.com:999999
SMTP server (HELO)	uni.fac.ehlo
SMTP server require a secure or encrypted connection	Yes / No
Username and Password for the SMTP account (performed in two steps; password entered in quiet mode)	-smtpaccount@uni.fac.com -smtppass
Default email address to which notifications can be sent	username@uni.fac.com

1.2.2. Install manually

In order to install manually do the following essential three steps:

1) Install CPAN modules

In your terminal type:

```
$ sudo cpan
```

This will give the CPAN command-line, where you can type to install all modules at once or type install <module> for one module at a time.

```
cpan[1]> install IO::All Email::Sender Email::MIME Sys::Info Term::ReadKey Thread::Semaphore
Statistics::Distributions Math::Cephes Bio::TreeIO File::Copy File::Copy::Recursive
```

Alternatively, without root privileges, CPAN can be configured and modules installed in the user account.

2) Install required programs (with administrative privileges: *sudo*)

In your terminal type:

```
$ sudo apt-get install paml screen sendmail-bin
```

or;

```
$ sudo apt-get install paml screen
```

If email notification is not required, the *sendmail* utility installation (Table 3) can be discarded (see section **1.2.1.2**). In both cases, it is assumed that all required programs will be installed for the first time, otherwise their installation can also be discarded.

Alternatively, without *sudo*,

download and install *PAML* from http://abacus.gene.ucl.ac.uk/software/paml.html
and follow *PAML*'s instructions; and

download and install *screen* from https://www.gnu.org/software/screen/
and follow *screen*'s instructions; and

download and install *sendmail* from http://www.sendmail.com/sm/open_source/download/ , ftp://ftp.sendmail.org/pub/sendmail
and follow *sendmail*'s instructions.

3) Proceed to LMAP configuration.

See section **1.2.1.2.**

4) Configure $HOME/bin.

To make all programs and scripts located at $HOME/bin (i.e., ~/bin) available at any working directory, it is also necessary to change $HOME/.bashrc file.
Place the following line in $HOME/.bashrc and save.

```
export PATH=$PATH:~/bin/:.
```

(see also: http://askubuntu.com/questions/9848/what-are-path-and-bin-how-can-i-have-personal-scripts).

Finally, reopen your terminal so that changes may take effect.

2. GETTING STARTED

2.1. Preparing Input Files

The *codeml* program from the PAML package, requires 3 different files to be created for execution, the multiple sequence alignment (MSA) (section 2.1.1), the phylogenetic tree (PT) files (section 2.1.2) and the *codeml* control file – LMAP templates (section 2.1.3). The MSA and PT files must be prepared and identified with specific name formats, whereas the *codeml* control files were transformed into 9 essential templates and need only user inspection to ensure fit to his/her requirements.

2.1.1. Alignment files

[GENENAME]_[MODELTYPES][ICODE].[FILEEXT]

Where:

GENENAME – is any name or abbreviation given to the protein coding genes being analyzed. For instance, in case of mitochondrial encoded subunits cytochrome c oxidase III, this could be abbreviated as “COX3”.

MODELTYPES – is the specification of all the models that are required to be executed for this specific gene. For MODELTYPES see Table 5 below.

ICODE – the PAML specification of the *icode* parameter (Table 6), i.e. the translation table for the alignment.

FILEEXT – file extensions accepted for FASTA format can be one of “fas”, “fasta”, “fst” or “txt”.

In this case, the file name would be “COX3_bw0.fas”.

NOTE 3: The **underscore** character ('_') must be always present and no spaces should be found in the file names. If these conditions are not met, the alignment file(s) will be excluded from execution.

Table 5. *Codeml codon-substitution models.*

General Description	Abbr.	MODELTYPES	Model Abbreviations	Labeling
Site models	SM	s	M0, M1a, M2a, M3, M7, M8, M8a	No
Branch-site models	BSM	w	Null (MA1), Alternative (MA)	Yes – Branch related models (BRM)
Branch models	BM	b	M0, TrC, TrU	
Clade models	CM	c	M2a_rel, CmC	

Table 6. *Genetic codes as available from GenBank and according to PAML.*

Genetic codes (ICODE)	Description
0	Universal
1	Mammalian mitochondrial
2	Yeast mitochondrial
3	Mold mitochondrial
4	Invertebrate mitochondrial
5	Ciliate nuclear
6	Echinoderm mitochondrial
7	Euplotid mitochondrial
8	Alternative Yeast nuclear
9	Ascidian mitochondrial
10	Blepharisma nuclear

2.1.2. Phylogenetic trees

Depending on the MODELTYPE to run, there are two possible ways of identifying PTs:

2.1.2.1. The case of SM [non-BRM (no labeling)]

Here, the PT files must be named in agreement with the corresponding alignment files and in the same way.

[GENENAME]_[MODELTYPES].[FILEEXT]

Where:
GENENAME – is any name or abbreviation given to the protein coding genes being analyzed, matching the corresponding alignment file GENENAME.
MODELTYPES – in this case is always and only the letter 's'.
FILEEXT – file extensions accepted for NEWICK format can be any of “nwk”, “newick” or “txt”.

Thus in this case would be “COX3_s.nwk”.

2.1.2.2. The case of other models [BRM (labeling)]

[HYPOTHESIS]_[MODELTYPES].[FILEEXT]

Where:
HYPOTHESIS – reference of the hypothesis being tested inherent to the tree labeling (H).
MODELTYPES – can be 1 or a combination of the three BRM ('b', 'w' or 'c').
FILEEXT – file extensions accepted for NEWICK format can be any of “nwk”, “newick” or “txt”.

Thus in this case possible examples would be “H_bwc.nwk” or “H_wb”.nwk or “H_c.nwk”.

NOTE 4: The **underscore** characters ('_') shown in both cases (2.1.2.1 and 2.1.2.2), must be always present and no spaces should be found in the file names. If these conditions are not met, they will be excluded from execution.

NOTE 5: Although it is possible to have a PT identified for running a combination of BRM (which implies the labeling is common for all models), this is only possible, assuming that the user data and devised hypothesis permit such scenario. It is the user responsibility to understand the scientific viability of such tree labeling and identification.

For an explanation of how to select models and their relation to input file identities, please see sections 3.1 and 3.1.2.

2.1.3. *Codeml* control files – LMAP Templates

LMAP provides 9 preconfigured *codeml* control files which we call templates. They are organized as seen in the following table. Their purpose is to ensure simplicity when preparing the dataset to run in LMAP (Table 7). These templates are located in the folder "LMAPlib/templates/".

Table 7. *LMAP Templates.*

Description	MODELTYPES	Models Abbrev.	Template file
Site models	s	M0, M1a, M2a, M3, M7, M8	codeml_sitemodelsM012378.ctl
		M8a	codeml_sitemodelsM8a.ctl
Branch-Site models	w	Alternative (MA)	codeml_branchsitemodelsALT.ctl
		Null (MA1)	codeml_branchsitemodelsNULL.ctl
Branch models	b	TrU	codeml_branchmodelsTRU.ctl
		M0	codeml_branchmodelsM0.ctl
		TrC	codeml_branchmodelsTRC.ctl
Clade models	c	CmC	codeml_clademodelsCMC.ctl
		M2a_rel	codeml_clademodelsM2a_rel.ctl

For any MODELTYPES selection all related templates will be selected and employed. The only exception is the SM case, where the user has the possibility to select specific models. Hence, if the user chooses not to run M8a, then this template will not be used. This choice is made in the command-line through the option -m (see section 3 and 8).

Some parameters, such as '*seqfile*', '*treefile*', '*icode*', '*NSsites*', '*omega*' and '*kappa*', are automatically modified by the application gmap.pl. They have values similar to HTML tags ("<tag>") which will be replaced during its execution.

Remaining "untagged" parameters were defined to be as close as possible to a definite

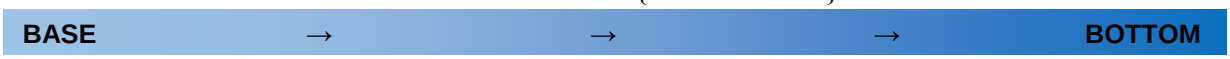
and correct configuration. Nevertheless, the user is encouraged to verify and modify any values to fit his/her requirements. Any changes, will remain applicable to any new LMAP execution, until new modifications take place.

After the templates are finally found to fit the researcher requirements, LMAP takes care of the distribution and final definitions that ensure the correctness of *codeml* executions.

2.2. LMAP Directory Structure

The directory structure created by LMAP, has the following profile from base to bottom:

LOCATION/PROJNAME/MODELTYPES/VALUES/{HYPOTHESES/}MODELNAMES/GENENAME



Where:

LOCATION – as given in option -d (see section 3 and 8). Where the project will be created and located. May contain several projects.

PROJNAME – as given in option -j (see section 3 and 8). The project name and the base of the directory structure. May contain at most 4 subfolders for each of the 4 MODELTYPES.

MODELTYPES – the specification of the set of models to be executed (Tables 5 and 7). May contain several subfolders leading to several executions of several omega or kappa values.

VALUES – depending on the MODELTYPES, these are the subfolders of omega and/or kappa values (options -O and -K – see section 3 and 8). Each value will contain the execution of several hypothesis or site models distributed in subfolders.

HYPOTHESES – Valid for the case of branch related models (BM, BSM and CM). For the case of SM, this level is absent from the directory structure. Each hypothesis will be executed for all *codeml* models here defined in subfolders.

MODELNAMES – the specific *codeml* models regarding the selected MODELTYPES. E.g.: For the case of BSM ('w'), this will consist of two distinct folders specifying null and alternate models. Each one will finally contain the subfolders regarding each MSA of the input dataset.

GENENAME – the tip of the directory structure, which ends with the GENENAME identification. All files regarding *codeml* execution (codeml.ctl, MSA and PT) are here located (i.e., in each folder identified after the corresponding MSA or GENENAME).

3. APPLICATION *gmap.pl* (version: 1.0.0 Nov 20th, 2015)

The gmap.pl is has the following main functions:

3.1) Create the directory structure necessary for *codeml* executions.
3.2) Label an already existing phylogenetic tree file to create different hypothesis.

To create the directory structure, the user needs the MSA files in FASTA format and the corresponding PT files in NEWICK format. These files can then be located in a unique folder or in two separate folders. To understand how these files should be prepared, see section **2.1**. To understand how the generated directory structure is organized, see section **2.2**.

3.1. Create the LMAP Directory Structure

Assuming all the input files are placed in the Data folder.
To create the directory structure simply type:

```
$ gmap.pl -A Data/MSA/ -T Data/Trees/ -d . -j MyDirectoryStruct -m b,w,s[0:1:2:3:7:8:8a],c
```

This command will retrieve all the necessary files, alignments (option -A) and phylogenetic trees (option -T), from the Data folder and create the directory structure named MyDirectoryStruct (option -j) in the current directory (option -d) prepared to run the selected models (option -m) and with the provided input files hereby organized in separate folders.

Here the option -m specifies that all models will run (BM, CM, BSM, and SM).

NOTE 6: the option for running SM is “s[0:1:2:3:7:8:8a]” which clearly specifies the execution of all SM (M0, M1a, M2a, M3, M8 and M8a) and consequently two templates will be used (see Table 7). The numbers indicate the required models to run (**0:** M0, **1:** M1a, **2:** M2a, etc.). Other correct possibilities for this would be “s[1:2:7:8]”, or “s[7:8]”, or “s[7:8:8a]”.

3.1.1. Adjusting initial *kappa* and *omega* values

Other two options may be specified giving the user the possibility to choose initial values to be specified in the *codeml* control files. These options are available to detect and avoid local optima [3]. For the case of BM and CM (Table 8), the user can adjust omega values through the option -O (upper case o). For the case of BSM and SM (Table 8), the user can adjust kappa values through the option -K (upper case k). In case these options are not specified, the default values prevail (see Table 8). For instance, the user can type either one of:

```
$ gmap.pl -A Data/MSA/ -T Data/Trees/ -d . -j MyDirectoryStruct -m b,w,s[0:1:2:3:7:8:8a],c -O
b[0.0,0.25]:c[0.5,0.8]
```

This case will only define omega values and kappa values will be default.

```
$ gmap.pl -A Data/MSA/ -T Data/Trees/ -d . -j MyDirectoryStruct -m b,w,s[0:1:2:3:7:8:8a],c -K w[0.01,0.05]:s[2.5,1.5]
```

This case will only define kappa values and omega values will be default.

```
$ gmap.pl -A Data/MSA/ -T Data/Trees/ -d . -j MyDirectoryStruct -m b,w,s[0:1:2:3:7:8:8a],c -O b[0.0,0.25]:c[0.5,0.8] -K w[0.01,0.05]:s[2.5,1.5]
```

This case will define both omega and kappa values for all four MODELTYPES *(b, c, w and s).*

The last command will generate contents for only the values 0.0 and 0.25 for BM; 0.5 and 0.8 for CM; 0.01 and 0.05 for BSM and 2.5 and 1.5 for SM.

Table 8. *LMAP MODELTYPES and default kappa/omega values.*

Description	MODELTYPES	Kappa (-K)	Omega (-O)
Site models	s	0.2 ; 2 ; 5	n.a.
Branch-Site models	w	0.2 ; 2 ; 5	n.a.
Branch models	b	n.a.	0.0 ; 0.1 ; 0.01 ; 0.001 ; 0.25 ; 0.5 ; 0.75 ; 1 ; 1.5 ; 2
Clade models	c	n.a.	0.0 ; 0.1 ; 0.01 ; 0.001 ; 0.25 ; 0.5 ; 0.75 ; 1 ; 1.5 ; 2

n.a. – not applicable.

The options (-O and -K) require that MODELTYPES are identified in the same way as seen before in Table 8, separated by colon (':') if more than one; and with its values specified within brackets '[ ]'. The default values (Table 8) will always be used in the lack of specification for some MODELTYPES. Examples:

```
$ gmap.pl -A Data/MSA/ -T Data/Trees/ -d . -j MyDirectoryStruct -m b,w,s[0:1:2:3:7:8:8a],c -O c[0.5,0.8]
```

```
$ gmap.pl -A Data/MSA/ -T Data/Trees/ -d . -j MyDirectoryStruct -m b,w,s[0:1:2:3:7:8:8a],c -K s[2.5,1.5]
```

These commands will make gmap.pl to use the default values for branch models (first case) and for branch-site models (second case).

3.1.2. Choice of models and relation to input files

Here we explain how the choice of models (MODELTYPES) is related to input file identities. Examples are given and limitations presented. This explanation requires that the user has previously read section **2.1**.
First, all the models/files are selected whenever the researcher uses the option -m. This is

the main criteria that allows the researcher to run any set of codon models regardless of the configurations made in the input file names. Second, assuming that the user selects to run all models (-m s[0:1:2:3:7:8:8a],b,c,w), then only input files having the same specified (-m) model choices in their names, are selected. Whatever the selection of models made in this option, the combination of MSA and trees is only ultimately dependent of the configurations given in input file names. For instance, if there are files not complying to the rules from section **2.1**, then they will not be selected. Hence, the researcher may have an MSA identified as ATP6_sbwc1.fas which is able to be selected for any case (SM, BM, BSM and CM) and two trees: (1) a SM tree ATP6_s.nwk (always named after the intended gene name) and (2) one (or more) hypotheses trees 2WA_b.nwk (and/or AH_b.nwk). In case of SM, the MSA will always be combined with the ATP6_s.nwk tree since it is the only one configured with the MODELTYPE 's' and has same gene name. On the other hand, any MSA configured with the MODELTYPE 'b' can be paired with the 2WA_b.nwk tree. In the given examples, the MSA will be combined with both trees. Suppose there is another MSA identified as ATP8_b1.fas, this MSA will only be combined with the 2WA_b.nwk or other trees that have the MODELTYPE 'b'; but it will not be combined with a tree named as AH_w.nwk or as ATP8_s.nwk. This means that to be able to pair an MSA and a tree, the same MODELTYPES (either 's', 'b', 'c', 'w') must be configured in both input files and they will be selected and executed provided that the user specifies them in option -m.

Although this strategy is advantageous, if the researcher is not aware he may enter in erroneous executions. Suppose that the BRM ('b', 'w,' 'c') are planned for execution, for two datasets placed in the same folder (for instance, dataset A with five trees and B with four trees). To avoid possible errors or time waste with unwanted combinations, it will be necessary to separate the two datasets in two different folders for them to run separately. This avoids that one MSA from dataset A is combined with a tree from the dataset B, or otherwise a MSA from dataset B combined with a tree from dataset A.

3.2. (Interactive Mode) Label a Phylogenetic Tree

This application also allows the user to modify and label rooted PT's. This is accomplished in an interactive fashion. Assuming all the files have been placed in the Data folder, to enable labeling and/or modifying PT's simply type:

```
$ gmap.pl -A Data/MSA/ -t mytree.nwk -d . -j MyDirectoryStruct -m b,w,s[0:1:2:3:7:8:8a],c
```

The user is required to type the same options as in the previous section, except the option -t (lower case T) which replaces option -T and requires the path to the NEWICK file to be edited. It enables the gmap.pl application to enter in interactive mode, presenting an interactive screen, where the user can visualize the tree and easily perform the labeling of nodes (Figure 1).

The PT is displayed in the form of a cladogram and designed with textual characters. The drawing of branch lengths is independently made of any information found in the tree file. Bootstraps and branch lengths are possible to be shown or hidden (see section **3.2.7**). The Ancestral nodes are presented in the form "AN", where N is any number. In the same way, taxa or Leaf nodes are presented in the form "LN", where N is any number. These numbers are the specific identifiers for each node, which are necessary for using the interactive commands.

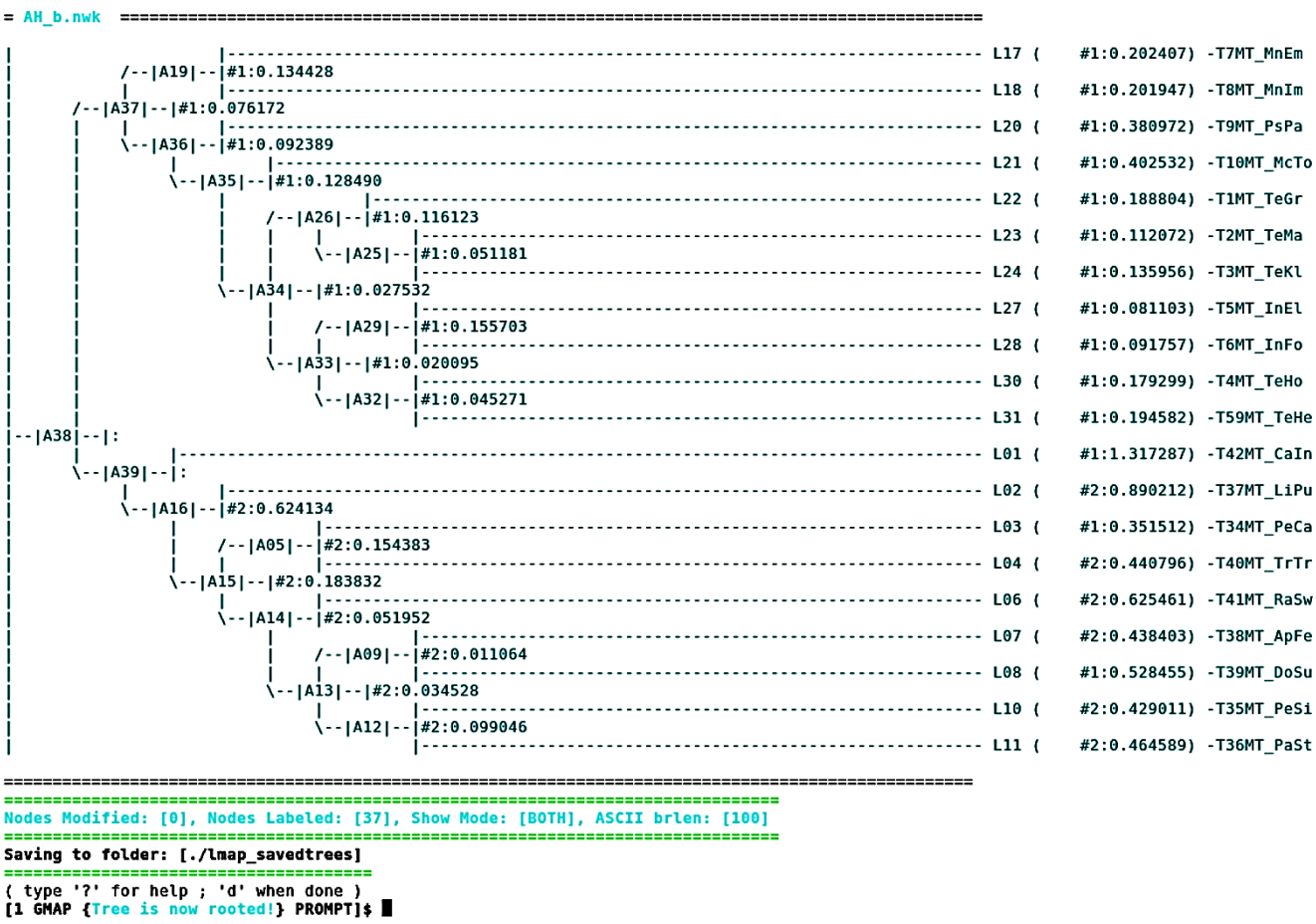


Figure 1. *gmap.pl interactive tree editing/labeling screen.*

This mode enables an interactive help menu provided whenever entering the question mark character ('?'). Here, the user has access to 14 interactive commands, with several functions (Figure 2). These are presented in the following sections.

Additionally, it is possible to go directly from labeling the trees to generating the directory structure. After the required tree editing is finished, the option -t will take to the option -T, hereby employing all the saved trees. This only depends on the user decision. If the decision is not to proceed, then gmap.pl will immediately terminate.

3.2.1. Choose location where to save phylogenetic trees

Phylogenetic tree files are by default saved in "./lmap_savedtrees/" folder created in the current directory. It is possible to change this location by using the command *'savetodir*'.

savetodir path_to_location

Where:
path_to_location – any location defined by a specific path.

```
        GMAP PROMPT HELP - COMMANDS
================================================================================

Options in [ ] - means, options are not mandatory; | - means, alternative options; ( ) - means, grouping

NOTE: The BioPerl modules here used, require that the inicial/input tree is rooted.
See http://www.bioperl.org/wiki/HOWTO:Trees

Command    Options          Description
========== ================ ======================
rbl                         Remove or clear branch length values, when available on the tree.
rbs                         Remove or clear bootstrap values, when available on the tree.
(Lx|Lx,...,Lz)=(#)y         Specify one or more species nodes (leafs) to be labeled with PAML symbols for branch (#),
                            followed by any required number eg.: L11=#1 or L07,L09,L11=#1.
(Ax|Ax,...,Az)=($|#)y       Specify one or more ancestral nodes to be labeled with PAML symbols for branch (#) or for clade($),
                            followed by any required number eg.: A11=#1 or A07,A09,A11=$1.
ls         [path|dir]       List contents on given directory. If no argument is given, lists current directory.
save       filename.nwk     Save tree with current configuration to filename.nwk. Default directory is ./lmap_savedtrees
saveunr    filename.nwk     Save rooted tree as unrooted with current configuration to filename.nwk. Default directory is ./lmap_savedtrees
savetodir  path|dir         Set default directory where to save (labeled) newick trees.
                            Will change default directory ./lmap_savedtrees to the one given as argument.
show       bs|bl|b|n        Show values/labels on the tree. Options are bs: bootstraps ; bl: branch-lengths ; b: both ; n:none.
brlen      n                Set branch length for display, n number of characters. By default is set to 100.
rootunr    Na,Nb            Root an unrooted tree, by specifying two nodes. This command creates a new node having Na and Nb as descendents
                            and the current 'root' as ancestor. This new node will hence be located below the current 'root' and above the specified descendents.
                            Nodes Na and Nb are any two nodes, leafs or ancestrals, connected to the current 'root'. eg.: rootunr A16,L01
                            Note: the word root was employed in quotes ('root'), since the node presented as most close to the left is not a true root.
                            In fact, by BioPerl design, every tree must be rooted and every unrooted tree has thus an additional root node,
                            which in this case becomes evident. Hence, the root displayed is in fact, a false root.
prune      (Nx|Nx,...,Nz)   Prune one or more nodes specified from phylogenetic tree, without changing the tree relationships.
                            Nodes can be tips (leafs) or ancestrals, in which case, the entire clade or descendent branches are removed,
                            including the specified ancestral node. eg.: prune A13 (ancestral) or prune L11 (leaf) or prune L11,L12,A17 (leafs and ancestral).
rtaxa      (Lx=N,...,Lz=N)  Rename phylogenetic tree taxa. This can be done with two alternative options: either by supplying an option
           |(file.csv)      consisting of 1 or more pairs of node/taxa (Lnumber) and associated new name (N), separated by commas, as in
                            (L01=Name1,L02=Name2,...); or an option indicating any titled CSV file listing two columns, containing at
                            the left column, the original name and at the right column, the new name.
                            eg.: rtaxa L11=Ted or rtaxa L11=Ed,L78=Tom,L80=Joe or rtaxa path/to/filename.csv.
reset                       Reread tree from file, thus forgetting any previous changes made.
---------- ---------------- ----------------------
d[one]                      Verifies if tree was saved, before exiting.
?                           This Help.
========== ================ ======================

VERSION: 1.0.0 Nov 20th, 2015
================================================================================

Type any command or press [ENTER] to go back to tree.

( type '?' for help ; 'd' when done )
[2 GMAP PROMPT]$
```

***Figure 2**. gmap.pl interactive commands help menu.*

3.2.2. Labeling operations

The labeling is a requirement of only the BRM and is performed according to the PAML manual specifications (#F and $F signs for branch and clades, respectively), where F is the number attributed to partition the foreground branches or clades. Labels are positioned as bootstraps values and are also displayed on screen (next to the associated nodes). Labeling can be performed both in Ancestral and in Leaf nodes.

For labeling Ancestral nodes, proceed as follows:

Ax,Ay,...,Az=#F
or;
Ax,Ay,...,Az=$F

For labeling Leaf nodes proceed as follows:

Lx,Ly,...,Lz=#F

Where:
Ax,Ay,...,Az – any ancestral (A) nodes, with x,y,...,z identifying any numbers presented on screen.
Lx,Ly,...,Lz – are any leaf (L) nodes, with x,y,...,z identifying any numbers presented on screen.
F – is any partition number.

Examples:
For labeling Nodes 5, 10 and 7, with “$1” just type:
A05,A10,A07=$1
For labeling Leaf Nodes 4, 8 and 12, with “#1” just type:
L04,L08,L12=#1

3.2.3. Root an unrooted tree

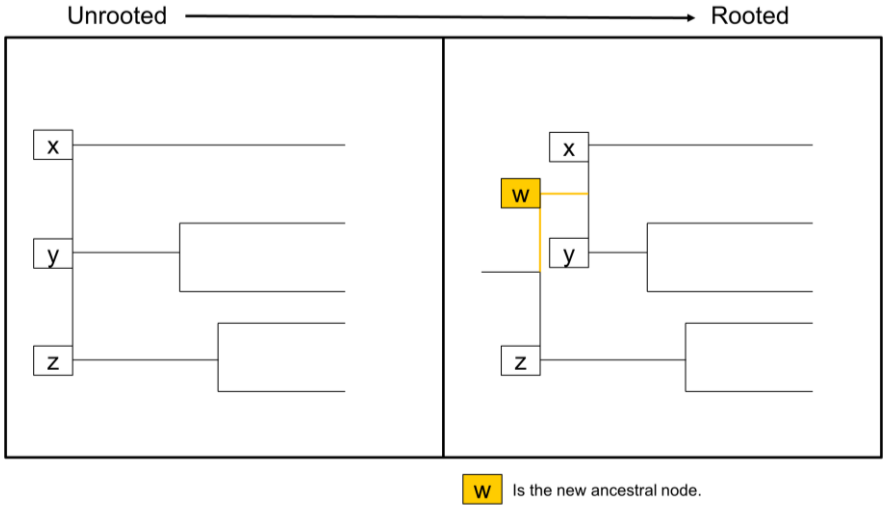


***Figure 3**. Scheme of the rooting operation.*

Sometimes it may be required to edit an unrooted PT. This requires the user to root the tree before it can be properly displayed and used.

In contrast to a rooted tree, a typical unrooted tree will have a basal trichotomy (http://mrbayes.sourceforge.net/Help/tree.html).
By BioPerl design, every tree must be rooted, whereas if an unrooted tree is displayed using gmap.pl, by default it will present an additional "root" node. To transform a trichotomy into a dichotomy a new ancestral node is created and inserted near the basal nodes (Figure 3).

The *'rootunr'* command must be used whenever the user needs to edit an unrooted tree. This will transform the tree into a dichotomy, through the specification of two basal nodes (x and y), which will serve as children of the new ancestral node (w) that this command will create.

rootunr Nx,My

Where:
Nx,My – are two ancestral (A) and/or leaf (L) nodes, with x,y identifying any numbers presented on screen.

Examples:
rootunr L01,A16
rootunr A37,L01
rootunr A16,A37

An example of the application of this command can be seen in Figure 1, where node A39 originated from specifying the nodes L01 and A16 as follows.

3.2.4. Save a phylogeny (as unrooted)

There are two ways to save a phylogeny: (i) one saves the phylogeny with the current configuration (command *'save'*); (ii) the other saves the phylogeny as an unrooted tree (command *'saveunr'*).

Both commands require the same arguments: the file name where to save the tree.

To save the rooted tree as unrooted (as required in *codeml* analyses), the user must select a file name and use the command *'saveunr'*.

save filename.nwk
or;
saveunr filename.nwk

Where:
filename – can be any file name for NEWICK format.

3.2.5. Modify *taxa* names

Taxa or Leaf names can be modified by using the command *'rtaxa'*. This can be applied either by using a file or directly in the command. Hence this can be done in two ways: (i) by creating a CSV file consisting of two columns ("Original/Current name" - left column) and ("Destination/New name" - right column) with any title as first row; or (ii) by using the

command alternative form, where at least one "LN=NewName" pair is required (one replacement). The first form may be more appropriate for many or all replacements, whereas the second may be more appropriate for casual modifications. This command has two ways of functioning:

rtaxa filename.csv
or;
rtaxa Lx=Name1,Ly=Name2,...,Lz=NameN

Where:
filename – can be any CSV file with two columns, where the left column, has the "old" names and the right column, the "new" names.
Lx,Ly,...,Lz – leaf nodes, with x,y,...,z identifying any numbers presented on screen.
Name1,Name2,…,NameN – new leaf identity to associate to each leaf node.

Examples:
rtaxa L11=Ned
rtaxa L11=Ned,L21=Red,L01=Ted,L24=Ed
rtaxa mynewtreenames.csv

3.2.6. Prune the phylogeny

There is also a command to prune the PT ('*prune*'), which allows the user to remove clades or branches, while keeping the original relationships.

prune Nx,Ny,...,Nz

Where:
Nx,Ny,…,Nz – can be ancestral (A) and/or leaf (L) nodes, with x,y,...,z identifying any numbers presented on screen.

Examples:
prune A10
- will remove the entire clade descending from ancestral node 10.

prune A10,L33
- will additionally remove the leaf node 33.

3.2.7. Removal and Display of bootstraps and/or branch lengths

Bootstrap and branch lengths values can be either erased from the tree or simply (un)hidden from the display.

To remove bootstrap values and branch lengths two commands without arguments are available: '*rbs*' to remove bootstrap values and '*rbl*' to remove branch lengths.

The display of these values on screen is managed by the command '*show*'. It requires one argument to allow control over the displayed values.

show v

Where:
v – is any value of the following: '*bs*' or '*bl*' or '*b*' or '*n*'. Respectively, they allow the display of bootstraps, branch lengths, both (i.e. 'bs' and 'bl') and none.

Examples:
show bs
show bl
show n
show b

The current display mode is monitored in the status bar below the tree cladogram under the tag "Show Mode" (Figure 1).

3.2.8. Other commands: reset, brlen, ls

Here the remaining simpler commands are described: *'reset'*, *'brlen'* and *'ls'*.
The command *'reset'* has the purpose of restoring the phylogenetic tree to its initial state. Careful is required since after this command, the node numbers are not the same.

The command *'brlen'* allows one to modify the length of the branches presented. It specifically affects the number of dashes (-) from the branches making them shorter or longer. By default, the length is set to 100.

brlen n

Where:
n – is a number specifying the length of the branches, in terms of the number of dashes.

This value is monitored in the status bar below the tree cladogram under the tag "ASCII brlen" (Figure 1).

The '*ls*' command is similar to the ls command from the terminal and allows the user to quickly see the directory contents without having to open a new terminal or exit gmap.pl. It lists the contents of any directory. It can be employed in two ways:

ls
or;
ls path_to_directory

Where:
path_to_directory – is any path to directory.

Without arguments, it lists the current directory (default behavior), otherwise lists any directory specified as usually performed in any terminal.

Examples:
ls /home/my_home/myworkdir/anotherfolder/

4. APPLICATION *cmap.pl* (version: 1.0.0 Nov 20th, 2015)

The cmap.pl application is useful when the user requires the change of any *codeml* control files parameters across a whole directory structure.
For instance, to the change the *icode* parameter value to 4 in every *codeml* control file occurring in the directory structure simply type:

```
$ cmap.pl -d MyDirectoryStruct -f e:ctl -g f:icode -L 4 -r
```

This command will change all the *codeml* control files found ending in "ctl" (option -f) in the whole (option -r) directory structure identified by the option -d, which have a parameter identified by its full name "icode" (option -f). In every file, this parameter value will be changed to the value 4 (option -L).

The option -f [Tag:Text] is appropriate to identify the control files to be modified through their names. Several alternatives are provided to adapt cmap.pl to various cases. This application can look for files depending on the Tag used; which are ending (tag "**e**"), starting (tag "**s**"), or having the full (tag "**f**") Text. Usually *codeml* control files are named as "codeml.ctl", and hence the option -f e:ctl is sufficient. This ensures that any files ending in "ctl" will be modified regardless of the file name before the dot.

The option -g [Tag:Parameter] is appropriate to identify the parameters to be modified in every control file. This option will make cmap.pl look for parameters which might end (tag "**e**"), start (tag "**s**"), contain (tag "**c**") or have the full (tag "**f**") Parameter text. Careful is required when handling with the tags **e**, **s** and **c**, which may affect several parameters. This can happen if the text for the parameter identification given after the colon (':') character, is common to other parameters. For instance, assuming the user wants to modify only the parameter "fix_alpha" the following could be done:

```
$ cmap.pl -d MyDirectoryStruct -f e:ctl -g s:fix -L 1 -r
```

(wrong!)

This command would affect all the parameters starting with the text "fix" given in option -g, as is the case of fix_kappa and fix_omega. Hence, all these parameters would be redefined with the value 1 (option -L). To properly modify only the parameter "fix_alpha" the tag **f** should be used and the full parameter name given, as in **"**-g f:fix_alpha".

```
$ cmap.pl -d MyDirectoryStruct -f e:ctl -g f:fix_alpha -L 1 -r
```

(right!)

In the other hand, this functionality can be useful if the user requires the modification of several parameters in one step.

NOTE 7: Although the **option -r**, is defined as not being mandatory, across the LMAP applications (cmap.pl, mmap.pl and imap.pl), it is required in most situations, whenever the user intends to affect the whole directory structure and not only the first folder. This option is always set by default for these applications, in lmap.pl.

5. APPLICATION *mmap.pl* (version: 1.0.0 Nov 20th, 2015)

The mmap.pl application has the purpose to execute and monitor all the available *codeml* control files existing in the directory structure.

To start the mmap.pl application simply type:

```
$ mmap.pl -d MyDirectoryStruct -r -n 30
```

This command will have mmap.pl application to start executing all *codeml* control files, found in the whole directory structure (option -r) given in the option -d. Furthermore, assuming that the workstation where LMAP is installed has at least a total of 30 CPU cores, this command will schedule a maximum number of 30 tasks (option -n) executing at same time.

The option -n is useful to control how many tasks will be running at a given time and to make possible the management of workstation capacity occupancy. The user can give a maximum number of 10 tasks and thus be able to execute two more mmap.pl instances with 10 tasks each, in the same workstation without any interference from the other instances. In fact, mmap.pl provides an ID (MMAPID or MPID) randomly generated. This MPID is composed of 2 consecutive capital letters and four digits as in “AZ0887”. This ID identifies all the screen instances (and hence *codeml* instances) executing with respect to the mmap.pl application which created them.

In a case, where multiple mmap.pl instances might be executing in the same workstation, each instance will show the tasks of other instances. Hence, to understand which is the mmap.pl instance responsible for some of the tasks, it is enough to observe the top left corner MPID (Figures 4-5) and compare to the MPIDs of the listed tasks. Additionally, and assuming the use of colors in LMAP is configured from the start (see section **1.2.1.2**), the tasks belonging to the specific mmap.pl instance will appear colored, contrarily to others.

Other options can be specified/added in the command line to adjust mmap.pl to specific cases. It is possible to adjust mmap.pl to select a different common (to all *codeml* control files) results file, by specifying the option -R (upper case r), as in “-R results.txt”. By default, the results file is “R”. Through a similar option -f [Tag:Text], it is possible to select *codeml* control files as explained in the section **4**.

Additionally, it is possible to provide a different source location for the *codeml* executable with the option -p [Tag:Program]. This requires the user to indicate a specific path (tag “**f**”), as in “-p f:/home/johndoe/software/paml/bin/codeml”. By default, it uses the default location chosen during LMAP configuration, which is accomplished through the tag “**l**” (lower case L), as in -p l:codeml.

The option -l (lower case L) enables the creation of a log file for reporting all *codeml* instances which have successfully finished. Here, the information is placed in three related columns, the first indicates the “rank” (numbering of which have finished first), the second indicates the *codeml* control “file” absolute path and the third, the “Time used”.

5.1. Monitoring of Executions and Available Screens

While mmap.pl is running, the user can monitor the *codeml* executions, which enables the user to understand the progress of the application and of the dataset. This monitoring of *codeml* tasks is performed in tasks that appear as scheduled, as running and as finished, in 2 different screens. The default screen 1 (Figure 4), shows pairs of "SCREEN" and "PRGM" lines. The "SCREEN" lines identify the *screen* utility, which is executing the following associated "PRGM" line. For instance, screen 13389 is executing *codeml* 13390 (Figure 4 – first two lines/numbers). Thus, each pair corresponds to one task. The task can be identified by the title shown in the "SCREEN" line. The title contains the *screen* process number, followed by the MPID which launched this task, and followed by the title, which consists in the full LMAP directory structure path to the location of the associated *codeml* task running. This location is represented with slashes converted to dashes.

```
[ MMAP MPID:TY8038, Started on: Tue May 17 20:42:39 2016, Current time: Tue May 17 20:52:19 2016 ]

RUN STATUS screen
====================
        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        ::: Status of the 32 instances running:
        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

SCREEN:         13389.MPID:TY8038:ExampleDatasetResults-SiteModels-0.2-M012378-ATP6     (17-05-2016 20:
   \_   PRGM:   13390 92.1  0.0 150132  9628 pts/13  [R: RUNNING]  20:42   8:54 ./codeml codeml.ctl
SCREEN:         13392.MPID:TY8038:ExampleDatasetResults-SiteModels-0.2-M012378-COX3     (17-05-2016 20:
   \_   PRGM:   13393 91.5  0.0 150352  9836 pts/23  [R: RUNNING]  20:42   8:50 ./codeml codeml.ctl
SCREEN:         13395.MPID:TY8038:ExampleDatasetResults-SiteModels-0.2-M8a-ATP6 (17-05-2016 20:42:39)
   \_   PRGM:   13396 92.8  0.0 150136  9624 pts/25  [R: RUNNING]  20:42   8:58 ./codeml codeml.ctl
SCREEN:         13398.MPID:TY8038:ExampleDatasetResults-SiteModels-0.2-M8a-COX3 (17-05-2016 20:42:39)
   \_   PRGM:   13399 92.1  0.0 150356  9844 pts/26  [R: RUNNING]  20:42   8:54 ./codeml codeml.ctl
SCREEN:         13401.MPID:TY8038:ExampleDatasetResults-SiteModels-2-M012378-ATP6       (17-05-2016 20:
   \_   PRGM:   13402 92.4  0.0 150136  9632 pts/27  [R: RUNNING]  20:42   8:56 ./codeml codeml.ctl
SCREEN:         13404.MPID:TY8038:ExampleDatasetResults-SiteModels-2-M012378-COX3       (17-05-2016 20:
   \_   PRGM:   13405 93.4  0.0 150356  9840 pts/28  [R: RUNNING]  20:42   9:01 ./codeml codeml.ctl
SCREEN:         13407.MPID:TY8038:ExampleDatasetResults-SiteModels-2-M8a-ATP6   (17-05-2016 20:42:39)
               ----------------------------------------//-------------------------------------

   \_   PRGM:   19925 97.3  0.0 154080 12496 pts/50  [R: RUNNING]  20:51   0:42 ./codeml codeml.ctl
SCREEN:         19927.MPID:TY8038:ExampleDatasetResults-BranchModels-0.0-AH-TrC-TMConc2 (17-05-2016 20:
   \_   PRGM:   19928 93.5  0.0 177264 35532 pts/52  [R: RUNNING]  20:51   0:41 ./codeml codeml.ctl
SCREEN:         20114.MPID:TY8038:ExampleDatasetResults-BranchModels-0.0-AH-TrU-ATP6     (17-05-2016 20:
   \_   PRGM:   20115 88.3  0.0 151264  9640 pts/51  [R: RUNNING]  20:51   0:23 ./codeml codeml.ctl
SCREEN:         20217.MPID:TY8038:ExampleDatasetResults-BranchModels-0.0-AH-TrU-ATP8     (17-05-2016 20:
   \_   PRGM:   20218 86.9  0.0 149828  8224 pts/49  [R: RUNNING]  20:52   0:09 ./codeml codeml.ctl
SCREEN:         20321.MPID:TY8038:ExampleDatasetResults-BranchModels-0.0-AH-TrU-COX1     (17-05-2016 20:
   \_   PRGM:   20324 87.0  0.0 153468 11532 pts/43  [R: RUNNING]  20:52   0:01 ./codeml codeml.ctl
SCREEN:         20323.MPID:TY8038:ExampleDatasetResults-BranchModels-0.0-AH-TrU-COX2     (17-05-2016 20:
   \_   PRGM:   20325 89.5  0.0 151296  9456 pts/53  [R: RUNNING]  20:52   0:01 ./codeml codeml.ctl

        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

                        Press s to show/hide other MPIDs.
              Press +/- to increase/decrease maximum number of instances running.

============================                                    ============================
  Press 2 for Task Status          VERSION: 1.0.0 Nov 20th, 2015        Ctrl-C or Ctrl-\ to exit...
============================                                    ============================
```

***Figure 4**. mmap.pl Screen 1 – Run Status screen.*

By pressing the keyboard key '2', mmap.pl displays two lists in screen 2 (Figure 5). At the top, the next 10 scheduled tasks are displayed ("Files running next"). These are tasks ready to enter execution, as soon as any task(s) from the screen 1 terminate(s). Below, the most recent 10 finished tasks are displayed ("Files finished") together with their "Time Used" at the left. The terminated tasks from screen 1, appear in this list, with the most

recent always showing at the top. Press keyboard key '1' to go back to screen 1 (Figure 4).

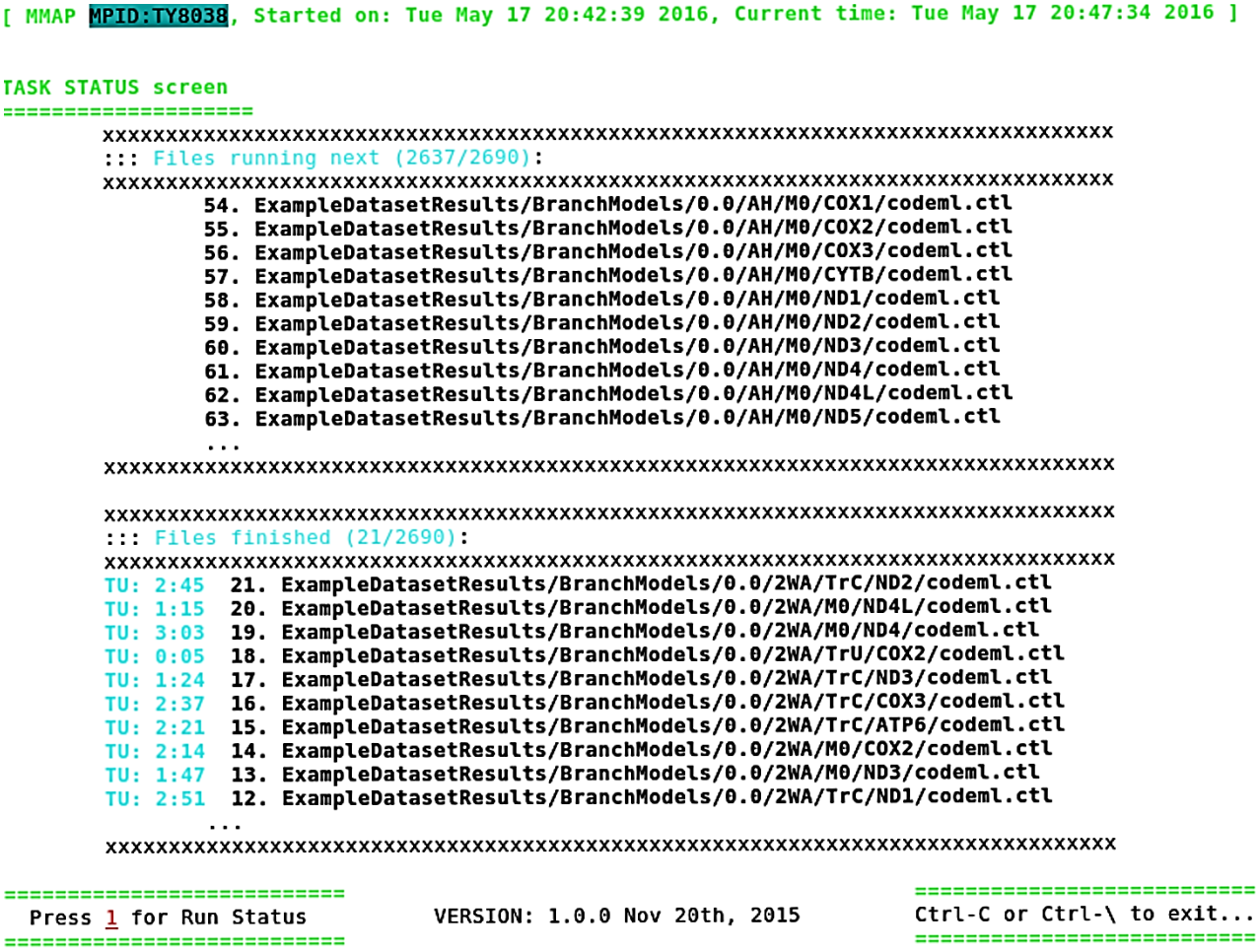


***Figure 5**. mmap.pl Screen 2 – Task Status screen.*

Beyond exhibiting all tasks, the screen 1 allows the user to understand if there is any *codeml* that is not running. This is displayed in the same column where the **"[R: RUNNING]**" tag is normally seen (see Figure 4). Any other tag (see Table 9) being exhibited in its place, indicates that something could be wrong with the input files specification. Additional log files are generated by the *screen* utility program in each folder of the LMAP directory structure. They contain the complete *codeml* output and aid in inspecting the source of the problems causing the *codeml* to behave differently, thus showing a different tag. In this case, the tag likely to occur more often would be the tag in the second row of Table 9. In this situation, the user may opt to terminate the task that is preventing another task to take its place. To this, the user may access the built-in Process Manager (PM) screen (Figure 6), by pressing "Ctrl-\" or "Ctrl-c" to exit. It will then be possible to enter this screen by replying "m" to the presented query.

***Table 9.** mmap.pl possible task status tags.*

Troubleshoot Tag	Observations
[R: RUNNING]	Execution OK.

[S: WAITING FOR USER INPUT]	Execution waiting for user input.
[T: STOPPED (BY SIGNAL)]	Execution stopped by external signal.
[D: UNINTERRUPTIBLE SLEEP (IO)]	Execution OK, but waiting for input/output access.
[Z: WRONGLY TERMINATED]	Execution terminated incorrectly.
[X: !DEAD!]	Should never be seen.

http://www.petefreitag.com/tools/man-pages/ps.html

NOTE 8: After typing “Ctrl-\” or “Ctrl-c” to exit, this action may not be instantly triggered, and so it may require a moment to be presented the possibility to correctly terminate mmap.pl execution together with all *screen* instances. Three queries are presented to the user: the first, allows one to select the PM screen or to quit; the second, confirms the quit decision and the third, requires the user to decide what action to take towards the running tasks. In this last case, three actions are possible, (i) to terminate only spawned/created instances (current MPID), (ii) to terminate every screen instance running (includes other mmap.pl instances – all available MPIDs); or (iii) leave all running and just exit mmap.pl.

```
[ MMAP MPID:TY8038, Started on: Tue May 17 20:42:39 2016, Current time: Tue May 17 20:46:17 2016 ]

PROCESS MANAGER screen - Group or Single?
=======================

        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        MMAPID          PROCID          FILE                                        (32)
        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        TY8038          15654           ExampleDatasetResults/BranchModels/0.0/2WA/TrU/ATP8/codeml.ctl
        TY8038          13395           ExampleDatasetResults/SiteModels/0.2/M8a/ATP6/codeml.ctl
        TY8038          13422           ExampleDatasetResults/SiteModels/5/M8a/COX3/codeml.ctl
        TY8038          13419           ExampleDatasetResults/SiteModels/5/M8a/ATP6/codeml.ctl
        TY8038          13461           ExampleDatasetResults/BranchModels/0.0/2WA/M0/TMConc2/codeml.ctl
        TY8038          15673           ExampleDatasetResults/BranchModels/0.0/2WA/TrU/ND2/codeml.ctl
        TY8038          14658           ExampleDatasetResults/BranchModels/0.0/2WA/TrC/ND2/codeml.ctl
        TY8038          15663           ExampleDatasetResults/BranchModels/0.0/2WA/TrU/COX3/codeml.ctl
        TY8038          13389           ExampleDatasetResults/SiteModels/0.2/M012378/ATP6/codeml.ctl
        TY8038          13401           ExampleDatasetResults/SiteModels/2/M012378/ATP6/codeml.ctl
        TY8038          13410           ExampleDatasetResults/SiteModels/2/M8a/COX3/codeml.ctl
        TY8038          13413           ExampleDatasetResults/SiteModels/5/M012378/ATP6/codeml.ctl
        TY8038          15645           ExampleDatasetResults/BranchModels/0.0/2WA/TrC/ND5/codeml.ctl
        TY8038          15657           ExampleDatasetResults/BranchModels/0.0/2WA/TrU/COX1/codeml.ctl
        TY8038          13416           ExampleDatasetResults/SiteModels/5/M012378/COX3/codeml.ctl
        TY8038          15648           ExampleDatasetResults/BranchModels/0.0/2WA/TrC/TMConc2/codeml.ctl
        TY8038          15651           ExampleDatasetResults/BranchModels/0.0/2WA/TrU/ATP6/codeml.ctl
        TY8038          13392           ExampleDatasetResults/SiteModels/0.2/M012378/COX3/codeml.ctl
        TY8038          15793           ExampleDatasetResults/BranchModels/0.0/2WA/TrU/TMConc2/codeml.ctl
        TY8038          14948           ExampleDatasetResults/BranchModels/0.0/2WA/TrC/ND4/codeml.ctl
        TY8038          13407           ExampleDatasetResults/SiteModels/2/M8a/ATP6/codeml.ctl
        TY8038          15676           ExampleDatasetResults/BranchModels/0.0/2WA/TrU/ND3/codeml.ctl
        TY8038          15726           ExampleDatasetResults/BranchModels/0.0/2WA/TrU/ND4L/codeml.ctl
        TY8038          13398           ExampleDatasetResults/SiteModels/0.2/M8a/COX3/codeml.ctl
        TY8038          15679           ExampleDatasetResults/BranchModels/0.0/2WA/TrU/ND4/codeml.ctl
        TY8038          13404           ExampleDatasetResults/SiteModels/2/M012378/COX3/codeml.ctl
        TY8038          15669           ExampleDatasetResults/BranchModels/0.0/2WA/TrU/ND1/codeml.ctl
        TY8038          13471           ExampleDatasetResults/BranchModels/0.0/2WA/TrC/COX1/codeml.ctl
        TY8038          15795           ExampleDatasetResults/BranchModels/0.0/AH/M0/ATP6/codeml.ctl
        TY8038          15666           ExampleDatasetResults/BranchModels/0.0/2WA/TrU/CYTB/codeml.ctl
        TY8038          15642           ExampleDatasetResults/BranchModels/0.0/2WA/TrC/ND4L/codeml.ctl
        TY8038          15755           ExampleDatasetResults/BranchModels/0.0/2WA/TrU/ND5/codeml.ctl
        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

        To terminate processess type 'G:MMAPID' for a Group or 'P:PROCID' for a specific Process (or [D]one):
        > 
```

Figure 6. *mmap.pl – Process Manager screen.*

The PM screen presents a table with 3 related columns (Figure 6), the “FILE” column shows all running tasks (*codeml* control files locations), and two columns identify two IDs: (i) the “MMAPID” column, which allows the user to terminate a group of tasks that have the same ID and (ii) the “PROCID” column, which allows one to terminate a single *codeml* task. By identifying the problematic execution, the user may here issue the command “P:PROCID” or “G:MMAPID” with the corresponding ID to terminate it (e.g.: “P:15654” or “G:TY8038”). After all is done, the user may return to the main screen 1 or 2, following the mmap.pl queries.

5.2. Re-Run Unfinished or Invalid *codeml* Tasks

During the execution of mmap.pl some *codeml* executions may turn out to have errors due to, for instance, erroneous naming of taxa. These executions will appear tagged as “!ERR:PAMLEXC!” in screen 2 (Figure 5), which will mean that something went wrong. Each folder of the directory structure, beyond containing input files and *codeml* control file, will additionally comprise a file named “screenlog.0”. This file has the output from the *codeml* execution for the current folder. The user may use this file to help troubleshoot the possible error. After all erroneous executions have been checked and any problems solved, the user can easily re-run all the unfinished tasks by just typing the following:

```
$ mmap.pl -d MyDirectoryStruct -r -n 30 -x
```

Since other *codeml* instances have finished executing, there is no need in re-executing these successful tasks. The option -x, with no arguments, ensures all and only the unfinished tasks will be rescheduled.

NOTE 9: All in all, a *codeml* task is determined as “unfinished” or “unsuccessful” whenever the LMAP applications are unable to find the “Time used” line in the end of the results file.

5.3. Email Notifications

After LMAP has been configured with the correct SMTP information (see Table 4 and section **1.2.1.2.**), the user is ready to receive notifications. Notifications are sent, as soon as, the mmap.pl application terminates the batch of *codeml* executions.
Email notification is enabled via option -e, as in:

```
$ mmap.pl -d MyDirectoryStruct -r -n 30 -e username@uni.fac.com
```

A default email address can be defined during LMAP configuration (Table 4), which will be used whenever the option -e is employed without arguments. If this default address was not initially defined, then an address is required at all times as shown above.
This option can thus function in two ways: (i) send notification to the default address (-e); or (ii) to the address specified in front of this option (-e <address>). Thus whenever required, the user has the possibility of specifying a different address. In case no address was supplied in either case (in the command-line and during initial configuration), no notification will be sent and including this option will have no effect.

6. APPLICATION *imap.pl* (version: 1.0.0 Nov 20th, 2015)

The purpose of the imap.pl application is to extract *codeml* maximum likelihood parameters estimates from results files. This information often consists of, log-likelihoods, omegas, kappas, tree_length, BEB, etc.

Several methods have been developed to conveniently extract this information from every MODELTYPES. The extraction is performed from an entire directory structure where the specific MODELTYPES were executed. To this end, the user must type the following:

```
$ imap.pl -d MyDirectoryStruct/SiteModels/ -r -s[0,1,2,3,7,8,8a] -o MySMresults.csv
```

(with brackets)

This command will extract information regarding all indicated SM (M0, M1a, M2a, M3, M7, M8 and M8a) through option -s and collect the parameters values in the designated CSV file name (option -o). The option -r will determine that the application will search the entire directory structure indicated in option -d.

The output CSV file is organized with models in rows and ML parameters in columns. Here the first column is the absolute path to a determined results file, which serves to elucidate the user of how the results were obtained and with which initial values. The corresponding parameters follow up in the next columns (see Figure 7).

```
 FILE : ExampleDatasetResults_BMresults.csv  SAVE TO : ATP8_ExDsResults.csv
```

Mark	Index	File	File	File	File	File	File	Model	np	treeLength	kappa	omegas	lnl
[ ]	1.	ExampleDatasetRes	BranchModels	0.0	2WA	M0	ATP6	M0	39	10.28257	6.30549	0.07387	-5603.955166
[ ]	2.	ExampleDatasetRes	BranchModels	0.0	2WA	M0	ATP8	M0	39	11.14237	8.22504	0.17981	-1425.256769
[ ]	3.	ExampleDatasetRes	BranchModels	0.0	2WA	M0	COX1	M0	39	11.36621	7.92002	0.01245	-9666.587796
[ ]	4.	ExampleDatasetRes	BranchModels	0.0	2WA	M0	COX2	M0	39	9.49450	7.18295	0.03314	-4575.306275
[ ]	5.	ExampleDatasetRes	BranchModels	0.0	2WA	M0	COX3	M0	39	9.75202	6.53289	0.02734	-5232.679400
[ ]	6.	ExampleDatasetRes	BranchModels	0.0	2WA	M0	CYTB	M0	39	10.52480	6.16017	0.03859	-8392.403829
[ ]	7.	ExampleDatasetRes	BranchModels	0.0	2WA	M0	ND1	M0	39	12.07953	5.80582	0.04777	-7700.122440
[ ]	8.	ExampleDatasetRes	BranchModels	0.0	2WA	M0	ND2	M0	39	11.98698	5.81919	0.06836	-8818.082650
[ ]	9.	ExampleDatasetRes	BranchModels	0.0	2WA	M0	ND3	M0	39	11.74320	5.37862	0.11107	-3230.550654
[ ]	10.	ExampleDatasetRes	BranchModels	0.0	2WA	M0	ND4	M0	39	12.21656	6.35746	0.05042	-11049.759762
[ ]	11.	ExampleDatasetRes	BranchModels	0.0	2WA	M0	ND4L	M0	39	12.87895	6.23749	0.06319	-2420.550387
[ ]	12.	ExampleDatasetRes	BranchModels	0.0	2WA	M0	ND5	M0	39	11.74685	5.88700	0.05826	-14433.988323
[ ]	13.	ExampleDatasetRes	BranchModels	0.0	2WA	M0	TMConc2	M0	39	10.05168	5.73793	0.05329	-84188.317980
[ ]	14.	ExampleDatasetRes	BranchModels	0.0	2WA	TrC	ATP6	TrC	40	10.76482	6.62693	w0 = 0.12608;w1 =...	-5692.731979
[ ]	15.	ExampleDatasetRes	BranchModels	0.0	2WA	TrC	ATP8	TrC	40	11.31259	8.47987	w0 = 0.05357;w1 =...	-1431.039299
[ ]	16.	ExampleDatasetRes	BranchModels	0.0	2WA	TrC	COX1	TrC	40	12.79398	8.52619	w0 = 0.00757;w1 =...	-9869.153592
[ ]	17.	ExampleDatasetRes	BranchModels	0.0	2WA	TrC	COX2	TrC	40	11.36724	8.27190	w0 = 0.11316;w1 =...	-4642.020907
[ ]	18.	ExampleDatasetRes	BranchModels	0.0	2WA	TrC	COX3	TrC	40	10.94581	7.17885	w0 = 0.01529;w1 =...	-5333.291659
[ ]	19.	ExampleDatasetRes	BranchModels	0.0	2WA	TrC	CYTB	TrC	40	12.12601	6.87817	w0 = 0.04245;w1 =...	-8518.550715
[ ]	20.	ExampleDatasetRes	BranchModels	0.0	2WA	TrC	ND1	TrC	40	13.46568	6.10179	w0 = 0.03220;w1 =...	-7782.613896
[ ]	21.	ExampleDatasetRes	BranchModels	0.0	2WA	TrC	ND2	TrC	40	13.22177	6.16102	w0 = 0.07924;w1 =...	-8925.894571
[ ]	22.	ExampleDatasetRes	BranchModels	0.0	2WA	TrC	ND3	TrC	40	12.73455	5.79000	w0 = 0.08429;w1 =...	-3264.291176
[ ]	23.	ExampleDatasetRes	BranchModels	0.0	2WA	TrC	ND4	TrC	40	13.47859	6.78163	w0 = 0.06578;w1 =...	-11229.924824
[ ]	24.	ExampleDatasetRes	BranchModels	0.0	2WA	TrC	ND4L	TrC	40	13.86277	6.52500	w0 = 0.09822;w1 =...	-2447.653812
[ ]	25.	ExampleDatasetRes	BranchModels	0.0	2WA	TrC	ND5	TrC	40	12.70806	6.19040	w0 = 0.10148;w1 =...	-14650.975911
[ ]	26.	ExampleDatasetRes	BranchModels	0.0	2WA	TrC	TMConc2	TrC	40	11.05193	6.16961	w0 = 0.06042;w1 =...	-85415.083592
[ ]	27.	ExampleDatasetRes	BranchModels	0.0	2WA	TrU	ATP6	TrU	41	10.36475	6.34529	w0 = 0.15118;w1 =...	-5602.374268
[ ]	28.	ExampleDatasetRes	BranchModels	0.0	2WA	TrU	ATP8	TrU	41	11.09575	8.21864	w0 = 0.06494;w1 =...	-1424.847165
[ ]	29.	ExampleDatasetRes	BranchModels	0.0	2WA	TrU	COX1	TrU	41	11.46714	7.95452	w0 = 0.01094;w1 =...	-9666.150703
[ ]	30.	ExampleDatasetRes	BranchModels	0.0	2WA	TrU	COX2	TrU	41	9.66350	0.00011	w0 = 0.00011;w1 =...	-10213.714684

```
   ...
================================================================================
[2340] rows, [13] columns, [0] rows marked, [30] rows showing, [0] columns hidden
================================================================================
SCROLLING  ON: Ctrl+d , OFF: Ctrl+\
Scroll Keys: Arrows [UP / DOWN], Top: [HOME], Bottom: [END]
================================================================
( type '?' for help ; 'q' to quit )
[1 U OMAP PROMPT]$
```

***Figure 7**. imap.pl retrieved data from BM (presented by omap.pl User Table).*

In Figure 7, columns named “File” compose the path to a specific *codeml* control results file. The following columns define the values retrieved, with origin in this associated path.

Other examples:

```
$ imap.pl -d MyDirectoryStruct/CladeModels/ -r -c -o MyCMresults.csv
```

(EXTRACT CM INFORMATION)

```
$ imap.pl -d MyDirectoryStruct/BranchSiteModels/ -r -w -o MyBSMresults.csv
```

(EXTRACT BSM INFORMATION)

```
$ imap.pl -d MyDirectoryStruct/BranchModels/ -r -b -o MyBMresults.csv
```

(EXTRACT BM INFORMATION)

Notice that in all four examples the location given in option -d, the PROJNAME subfolders are specified, thus pointing imap.pl to the extraction of the corresponding models according to the options indicated (-s, -c, -w and -b). This is necessary to ensure that imap.pl correctly extracts information for each case at a time (Table 10).

Table 10. *imap.pl options for extraction of information for each MODELTYPE.*

Description	**MODELTYPES**	**imap.pl** options	**PROJNAME** subfolders
Site models	s	-s[models]	PROJNAME/SiteModels/
Branch-Site models	w	-w	PROJNAME/BranchSiteModels/
Branch models	b	-b	PROJNAME/BranchModels/
Clade models	c	-c	PROJNAME/CladeModels/

The following is also possible given that there is only one case under the PROJNAME.

```
$ imap.pl -d MyDirectoryStruct/ -r -w -o MyBSMresults.csv
```

(EXTRACT BSM INFORMATION)

Hence, whenever extracting information from all the MODELTYPES, imap.pl must be run four times.

Besides the CSV output generated, imap.pl creates two additional files, that may aid the user to verify if all executions have correctly finished. These two files list (i) the correctly finished tasks and (ii) the incorrectly finished, if any.

7. APPLICATION *omap.pl* (version: 1.0.0 Nov 20th, 2015)

This application takes as input the CSV file obtained from the imap.pl application to organize and perform LRT estimates. Contrarily, to the gmap.pl application, omap.pl is an entirely interactive application. It only requires the user to specify the input (option -i) and output file (option -o) in the command-line:

```
$ omap.pl -i MyBSMresults.csv -o MyBSM_LRTresults.csv
```

During loading of the input CSV file, the existing BEB column will be stored separately, to allow the LRT estimations to take place.

It implements a total of 24 interactive commands (Figure 8) to perform several operations over the imap.pl results. Here two containers are implemented to the help the user organize the input data: (i) the default container, named “User Table” (U), initiates with the data loaded from the input file (option -i); while (ii) the second container, named “Final Table” (F), was designed to finalize data organization and to finally make the LRT estimations. Beyond the 24 interactive commands, omap.pl also enables table scrolling. As seen in figures 7 and 9, scrolling is activated (ON) simply by typing the keyboard combination “Ctrl+d”. Hereafter, up and down arrow keys can be used to scroll the table row-by-row up and down. The keys “Home” and “End” scroll the table directly to the top and to the bottom, respectively. To be able to type any interactive commands, scrolling must be deactivated (OFF), which is accomplished by typing the keyboard combination “Ctrl+\” (control + backslash).

7.1. Switch Tables / Change Number of Visible Rows

To allow the user switch among containers the command *’show’* is employed. This command can be employed in three ways:

show T
or;
show y
or;
show T y

Where:
T – is any of the containers specified by the letter 'f' (F) or 'u' (U).
y – is any number that allows the user to adjust the number of rows currently displayed.

Examples:

```
show f
show 50
show u 20
```

With this command it is possible to (i) switch containers/tables, (ii) adjust the visible number of rows or (iii) even do both.

```
        OMAP PROMPT HELP - COMMANDS
================================================================================

Options in [ ] - means, options are not mandatory; | - means, alternative options; ( ) - means, grouping

Command    Options         Description
========== =============== =====================
open       filename.csv    (Only in User Table! see command show) Open CSV file, discarding any current values.
ls         [path|dir]      List contents on given directory. If no option is given, lists current directory.
reset                      Reset contents in User Table to initial contents.
delcl(*)   C(a|a-z|a,d,h)  Remove column(s). Either a single column or a range of columns specififed by from-to values or any columns comma-separated.
hide       C(a|a-z|a,d,h)  (Un)Hide column(s). Either a single column or a range of columns specififed by from-to values or any columns comma-separated.
sort(*)    Cn (A|a)|(D|d)  Sort column C[number] [A|a]scending or [D|d]escending. This is performed according to its contents type.
show(*)    (n|t)|(n t)     Show i) (number of rows n) or (Table t) or ii) number of rows (n) in Table (t). (t = [U|u]ser or [F|f]inal).
mark(*)    [a|a-z|a,d,h]   (Un)Mark rows. Either a single row, a range of rows specififed by from-to values or any rows comma-separated;
                           If no option is given, the selection will (un)mark all rows in Table.
markf(*)   r Cn [l]        (Un)Mark the first r rows which contain distinct values in column C[number];
                           Search within l rows limit or, by default, in full Table length.
findr(*)   "value" Cn      (Un)Mark all rows containing the quoted specified value in column C[number];
                           The characters '^','$','|','[' and ']', can be used in the value to mean either 'start', 'end', 'or', or 'set of alternative characters', respectively;
                           eg.: findr "^value" C4 ; findr "value$" C4 ; findr "val1|val2|val3" C4 ; findr "[vV]alue" C4.
copy(*)    [n]             (Only in User Table! see command show) Copy any marked rows, n times each, from User Table to Final Table. If not given n, number of copies defaults to 1;
                           Copies are differentiated by 'index.subindex' in column 'Index', 2nd from the left. Use command 'findr' to (un)mark any copies.
move(*)                    (Only in User Table! see command show) Move any marked rows from User Table to Final Table.
mrup(*)    [n]             Move any marked rows n positions up in current Table. If not given n, number of positions defaults to 1.
mrdn(*)    [n]             Move any marked rows n positions down in current Table. If not given n, number of positions defaults to 1.
delrs(*)                   Delete any marked rows in current Table.
fh                         Show file headers in current Table.
empty(*)                   Empty Final Table contents.
plrt(*)    significance    (Only in Final Table! see command show) Perform LRT test for every pair of rows, in Final Table, from top to bottom;
                           The null model should be placed on the row above the row of the alternative model. In case rows are marked, this operation is performed preferencially in the selection.
pbeb(*)                    (Only in Final Table! see command show) Create an additional column containing the BEB information considering the LRT test conclusion if exists;
                           otherwise, BEB will appear in all rows regardless of the LRT conclusion. This column can only be placed, if initially existed in the input file.
save[!](*) [file.csv]      (Only in Final Table! see command show) Save Final Table contents to default output file (from initial option -o) or if specified, to 'file.csv';
                           Append exclamation mark at the end of command, as in 'save! file.csv', to save contents and make the specified CSV file the default;
                           The contents will be appended to the end of the default file, whenever the file exists; regardless if the file is empty or is a new file;
                           Only marked rows (see command mark, markf, findr), are saved whenever found in Table; or otherwise, all is saved.
def        [CMD[=LIST]]    Define an alias command to perform several operations at once; two other uses can take place:
                           1) The user chooses a command name (CMD) (not found in this HELP) and a LIST of commands, where LIST=(cmd1;...;cmdn), following previous rules for each command;
                           2) Type 'def CMD' to show the definition for the specified command;
                           3) Type 'def' only to show the list of all the defined commands;
                           (*) The commands allowed in the LIST are here marked by an asterisc.
defsave    [file.name]     Save all defined commands to file. File is by default /home/labpc3c/.omapd, if none is given.
defload    [file.name]     Load all defined commands from file. File is by default /home/labpc3c/.omapd, if none is given.
defdel                     Delete /home/labpc3c/.omapd file.
                           This file may contain several definitions including redefinitions of the same command. Hence, the last definitions in file prevail.
---------- --------------- ---------------------
q[uit]                     Checks if any unsaved data are there in Final Table, before exiting.
?                          This Help.
========== =============== =====================

VERSION: 1.0.0 Nov 20th, 2015
================================================================================

Type any command or press [ENTER] to go back to Table.

( type '?' for help ; 'q' to quit )
[1 U OMAP PROMPT]$
```

***Figure 8**. omap.pl interactive commands help menu.*

7.2. (De)Select Rows

The commands *'mark'*, *'markf'* and *'findr'* all have the same purpose, which is to *select* and *deselect* rows from the table (either U or F). These commands have a special behavior, which is dependent on the current selection found in the table. In one run, the selected rows will become deselected and any deselected rows will become selected. The difference between them lies in how they achieve this.

7.2.1. The *mark* command

This command is the most straightforward and can be employed in three ways:

mark
or;
mark x-y
or;
mark x,y,z,a,b

Where:
a,b,x,y,z – are any row numbers presented on the table (see "Index" column – Figure 7).

Examples:
mark
mark 10-25
mark 1,8,2,7,5,10,12,13

The first case, with no arguments, affects the entire table by inverting any (de)selection. The second, the (de)selection is limited to the range of rows specified from x to y (where, x < y). The third case, gives the freedom to (de)select any random rows.

7.2.2. The *markf* command

The command *'markf'* will search a column of the table for the first occurrences of a specified number of different values, to (de)select the corresponding rows:

markf r Cn [z]

Where:
r – the specified number of occurrences of different values.
Cn – the specified Column number where to search.
z – search within z rows limit (optional).

Examples:
markf 3 C7
markf 3 C7 40

This command will search the column Cn top-down and will stop as soon as it finds r different values whose rows will be finally (de)selected. For instance, if the column has the values x and y several times repeated and if r = 2, then the first two rows where the values

x and y, are found (through the given column), will be (de)selected. Additionally, a third optional argument can be specified to limit the search to z rows.

7.2.3. The *findr* command

During a column search, the command *'findr'* will (de)select all rows, where a given value is found.

findr "r" Cn

Where:
r – the value to search for. Always in quotes "".
Cn – the specified Column number where to perform the search.

Examples:
findr "M0" C6
findr "TrU" C6
findr "M0|TrU" C6

Special characters can be additionally specified in quotes along with the value, to help select the required matching rows.
The character '^' placed before the value ensures that the matching text found, must start by the following value characters, as in "^value". As opposite, the character '$' placed after the value, ensures that the text found, must end by the previous value characters, as in "value$". The pipe character '|' can be employed several times to mean alternatives, as in "value1|value2|value3", allowing to search several alternative values in one go. The bracket characters "[]", allow the user to place alternatives to specific positional characters, ensuring for instance case-insensitiveness, as in "[vV]alue". This means that it will search for text either starting with "value" or with "Value" and thus (de)select all matching rows.

7.3. Move Rows

The move commands, (*'move'*, *'mrup'* and *'mrdn'*), allow the user to move any selected rows. The first moves rows from table U to F (no argument is required). The second and third, move rows in the same container, up (*mrup*) or down (*mrdn*). By default, these two move rows only one position/row, but it is possible to specify a number as argument to move rows more than one position, e.g.:

move
or;
mrup [n]
or;
mrdn [n]

Where:
n – the number of row positions to move, by default this is 1.

Examples:
mrup
mrdn

mrup 3
mrdn 2

The third example will move any selected rows 3 positions up. The fourth will move any selected rows 2 positions down. The set of rows moved, will not become contiguous or adjacent at the end. This means, for instance, that if rows 2 and 4 are selected and if the third command is applied, row 2 will be moved up 3 positions as will row 4, thus keeping the same distance between the two selected rows.

7.4. Sort Rows

The *'sort'* command allows one to sort the rows of any table according a specified column contents and required order.
This command is applied as follows:

sort Cn r

Where:
Cn – the specified Column number to use for sorting (e.g.: C8).
r – the sort order: either ascending (value 'A' or 'a') or descending ('D' or 'd').

Examples:
sort C5 a
sort C5 d
sort C5 A
sort C5 D

Each of these examples sorts the table rows through column C5 ascending (1st and 3rd) or descending (2nd or 4th). In any case, the sort is automatically accomplished according to the column content type. That is, if the column has values that are entirely numbers, the sort is numerical, otherwise the sort is alphabetical.

7.5. Copy Rows

The command *'copy'* allows a user to perform copies of selected rows from the table U to table F.

copy [n]

Where:
n – the number of copies, by default this is 1.

Examples:
copy
copy 3

When used with no arguments, makes one copy of the selected rows. Multiple copies can be performed, by using the optional argument.

7.6. Delete Columns/Rows

To delete rows, the command *'delrs'* will delete any selected rows from the current table (no argument is required). To delete columns, the command *'delcl'* will delete any specified columns in two ways:

delrs
or;
delcl Cx-y
or;
delcl Cx,y,z

Where:
x,y,z – any specified column numbers to delete. Always indicated after the capital letter "C".

Examples:
delrs
delcl C5-10
delcl C7,8,3,9,12

The command *delcl* allows one to delete a range of columns (first case) from x to y (where, x < y) or any random columns (one or more), in the second case.

7.7. Hide Columns

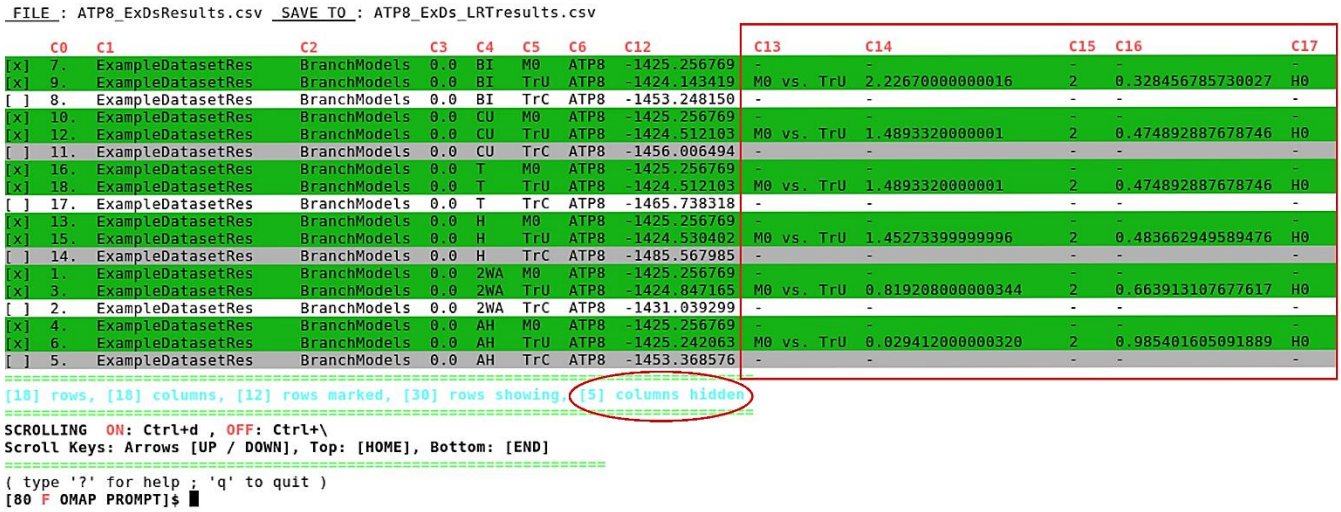
Sometimes it is useful to reduce the space on the screen for better visualization of the table. This would require to remove columns, but then necessary information would be lost.
The '*hide*' command solves this purpose and hides any required columns from the table view. The number of columns hidden, is shown in the status bar below the table (Figures 7 and 9 – red oval). Similarly, to the previous command, it works in two ways:

hide Cx-y
or;
hide Cx,y,z

Where:
x,y,z – any specified column numbers to delete. Always indicated after the capital letter "C".

Examples:
hide C5-10
hide C6,7,9,3

This command allows one to hide a range of columns (first case) from x to y (where, x < y) or any random columns (one or more), in the second case.



FILE : ATP8_ExDsResults.csv SAVE TO : ATP8_ExDs_LRTresults.csv

	C0	C1	C2	C3	C4	C5	C6	C12	C13	C14	C15	C16	C17
[x]	7.	ExampleDatasetRes	BranchModels	0.0	BI	M0	ATP8	-1425.256769	-	-	-	-	-
[x]	9.	ExampleDatasetRes	BranchModels	0.0	BI	TrU	ATP8	-1424.143419	M0 vs. TrU	2.22670000000016	2	0.328456785730027	H0
[ ]	8.	ExampleDatasetRes	BranchModels	0.0	BI	TrC	ATP8	-1453.248150	-	-	-	-	-
[x]	10.	ExampleDatasetRes	BranchModels	0.0	CU	M0	ATP8	-1425.256769	-	-	-	-	-
[x]	12.	ExampleDatasetRes	BranchModels	0.0	CU	TrU	ATP8	-1424.512103	M0 vs. TrU	1.4893320000001	2	0.474892887678746	H0
[ ]	11.	ExampleDatasetRes	BranchModels	0.0	CU	TrC	ATP8	-1456.006494	-	-	-	-	-
[x]	16.	ExampleDatasetRes	BranchModels	0.0	T	M0	ATP8	-1425.256769	-	-	-	-	-
[x]	18.	ExampleDatasetRes	BranchModels	0.0	T	TrU	ATP8	-1424.512103	M0 vs. TrU	1.4893320000001	2	0.474892887678746	H0
[ ]	17.	ExampleDatasetRes	BranchModels	0.0	T	TrC	ATP8	-1465.738318	-	-	-	-	-
[x]	13.	ExampleDatasetRes	BranchModels	0.0	H	M0	ATP8	-1425.256769	-	-	-	-	-
[x]	15.	ExampleDatasetRes	BranchModels	0.0	H	TrU	ATP8	-1424.530402	M0 vs. TrU	1.45273399999996	2	0.483662949589476	H0
[ ]	14.	ExampleDatasetRes	BranchModels	0.0	H	TrC	ATP8	-1485.567985	-	-	-	-	-
[x]	1.	ExampleDatasetRes	BranchModels	0.0	2WA	M0	ATP8	-1425.256769	-	-	-	-	-
[x]	3.	ExampleDatasetRes	BranchModels	0.0	2WA	TrU	ATP8	-1424.847165	M0 vs. TrU	0.819208000000344	2	0.663913107677617	H0
[ ]	2.	ExampleDatasetRes	BranchModels	0.0	2WA	TrC	ATP8	-1431.039299	-	-	-	-	-
[x]	4.	ExampleDatasetRes	BranchModels	0.0	AH	M0	ATP8	-1425.256769	-	-	-	-	-
[x]	6.	ExampleDatasetRes	BranchModels	0.0	AH	TrU	ATP8	-1425.242063	M0 vs. TrU	0.029412000000320	2	0.985401605091889	H0
[ ]	5.	ExampleDatasetRes	BranchModels	0.0	AH	TrC	ATP8	-1453.368576	-	-	-	-	-

[18] rows, [18] columns, [12] rows marked, [30] rows showing, [5] columns hidden

SCROLLING ON: Ctrl+d , OFF: Ctrl+\
Scroll Keys: Arrows [UP / DOWN], Top: [HOME], Bottom: [END]

(type '?' for help ; 'q' to quit)
[80 F OMAP PROMPT]$

***Figure 9**. omap.pl Final Table showing the LRT estimations.*

7.8. Perform LRTs and BEB: plrt, pbeb

Since the selected rows are moved to the Final Table, the LRT estimations can be performed. Here the rows of null and alternative models counterparts must be grouped together, with the null model above the alternative model row (Figure 9). The command to estimate the LRTs among null and alternative models is *'plrt'*. After the estimations, whenever there is a BEB column, it can be placed back possibly according to the LRT significance. The command for this is *'pbeb'*.

These commands are employed as follows:

plrt x
or;
pbeb

Where:
x – any specified decimal confidence value - confidence.

Examples:
plrt 0.05
pbeb

Depending on whether rows are selected, the *'plrt'* command works in two ways: (i) in case none of the rows is selected, the LRT estimations are performed in the entire table; or (ii) otherwise the LRTs will be performed only in the selected rows. In this case, the rows of null and alternative counterparts must be selected in a pairwise fashion as in Figure 9. In either case, contents must obey the above mentioned grouping criteria.

The '*plrt*' command will insert five additional columns (C13-C17) in the table (Figure 9) each one informing about, (i) the LRT comparison performed (entitled *LRTtest*), (ii) the difference in log-likelihoods (entitled *deltaLnL*), (iii) the degrees of freedom (entitled *df*), (iv) the p-value (entitled *pvalue*) and (v) the conclusion (entitled *conclusion(confidence)*). The

values used for the estimations are from the grouped null and alternative counterparts, whose comparison is described in the *LRTtest* column. The conclusion column allows only two alternative values: H0, for the selection of the null hypothesis or H1, for the alternative hypothesis.

The *'pbeb'* command allows the restitution of the BEB information initially separated from the input data, through the insertion of an additional BEB column after the last LRT column (entitled *BEBsites*). This command requires no argument and has two different behaviors: (i) when issued after the LRT estimations (*'plrt'* command), the corresponding BEB information will be placed in every row where the conclusion column has the value H1, thus considering the significance given by the LRT estimation; otherwise, (ii) in case no LRT estimations were performed, the BEB information will be restituted to every available alternative model row.

This command will only operate in tables which initially, had the BEB column. Thus it will not work in BM results tables.

7.9. Save Table

The *'save'* command, saves the contents of the Final Table to file.
This command can be used in three ways:

save
or;
save filename.csv
or;
save! filename.csv

Where:
filename – any selected file name for CSV format.

Examples:

```
save
save ATP8_ExDs_LRTresults.csv
save! ATP8_ExDs_LRTresults.csv
```

The first form, by default saves the contents to the file specified in command-line (option –o). The second form allows the user to save to any different file, while the third form (with the extra exclamation mark), additionally makes the required file as default (thus replacing the one initially specified with option -o).

The *'save'* command allows contents to be saved selectively in two ways: (i) any rows found selected in table will be saved to file (ignoring the ones not selected); or otherwise (ii) in case none is selected, all table contents will be saved.

Nevertheless, the contents will be always appended to the file, provided that it exists. If it does not exist, it will be created and contents saved. In practice, the user may choose to use the output file to continuously save any data, or otherwise save different data in separate files.

7.10. Other commands: reset, fh, empty, open, ls

Here we describe the remaining simpler commands. The first three require no arguments.

The *'reset'* command is available to the table U, for restoring the initial contents.
The *'fh'* command is available to both U and F tables, to show the original file columns titles.
The *'empty'* command is available to both U and F tables, to empty the table F.

The *'open'* command allows the user to open a new CSV file to be loaded in initial table U. This command replaces the behavior of having to exit omap.pl and again re-executing the application just to load a new file.

open filename.csv

Where:
filename – any selected file name for the CSV format.

The '*ls*' command is similar to the ls command from the terminal and allows the user to quickly see the directory contents without having to open a new terminal or exit omap.pl. It lists the contents of any directory. It can be employed in two ways:

ls
or;
ls path_to_directory

Where:
path_to_directory – is any path to directory.

Without arguments, it lists the current directory (default behavior), otherwise lists any directory specified as usually performed in any terminal.

Examples:

```
fh
reset
empty
open ATP8_ExDsResults.csv
ls
ls /home/my_home/myworkdir/anotherfolder/
```

7.11. Def-related Commands (or How to Build My Own Commands?)

This section is dedicated to explain how to use the existing predefined commands seen so far to create additional commands, that can perform several operations just by typing a single interactive command.
The def-related commands are *'def'*, *'defsave'*, *'defload'* and *'defdel'*.
The first command has three functionalities depending on how it is employed:

def
or;
def mycmd
or;
def mycmd=x;y;z

Where:
mycmd – any name to define and ultimately call the new command.
x,y,z – any predefined commands (with their arguments) placed in any logical order, separated by semicolons.

Examples:
def
def f=show f
def u=show u;empty
def lrtbeb=plrt 0.05;pbeb
def finalize=plrt 0.05;pbeb;save;show u;empty

The first form simply lists all commands that have been defined so far. The second form shows how the command mycmd is defined. The third form, allows the definition of a new command. Here, a name must be given (mycmd) and its definition, which consists of a list of any commands selected by the user to be executed in order, from left (x) to right (z). This list can comprise any allowed commands. The allowed commands are marked with an asterisk (*) in the help menu and shown in the table below (Figure 8 and Table 11).

Table 11. *Commands allowed in the new command definition list.*

Commands allowed	Arguments	Section Reference
show	(n|t)|(n t)	see section 7.1
mark	[a|a-z|a,d,h]	see section 7.2.1
markf	r Cn [l]	see section 7.2.2
findr	"value" Cn	see section 7.2.3
move		see section 7.3
mrup	[n]	see section 7.3
mrdn	[n]	see section 7.3
sort	Cn (A|a)|(D|d)	see section 7.4
copy	[n]	see section 7.5
delcl	C(a|a-z|a,d,h)	see section 7.6
delrs		see section 7.6
plrt	confidence	see section 7.8
pbeb		see section 7.8
save[!]	[file.csv]	see section 7.9
empty		see section 7.10

The commands *defsave*, *defload* and *defdel* with no arguments, by default work with the default file located in the user home directory named ($HOME/.omapd).
Both commands *defsave* and *defload* respectively, save and load the defined commands to and from the default file. These additionally enable the user to save his/her commands to a different file of his/her choice. The command *defdel* deletes the default file (requires no argument).

defdel
or;
defsave
or;
defsave filename.txt
or;
defload
or;
defload filename.txt

Where:
filename – any selected file name for regular text file format.

8. APPLICATION *lmap.pl* (version: 1.0.0 Nov 20th, 2015)

The lmap.pl application has the purpose to further simplify the use and employment of the described applications. It enables their incorporation in a predefined order to accomplish in simplified manner the intended *codeml* workflow (Figure 10).

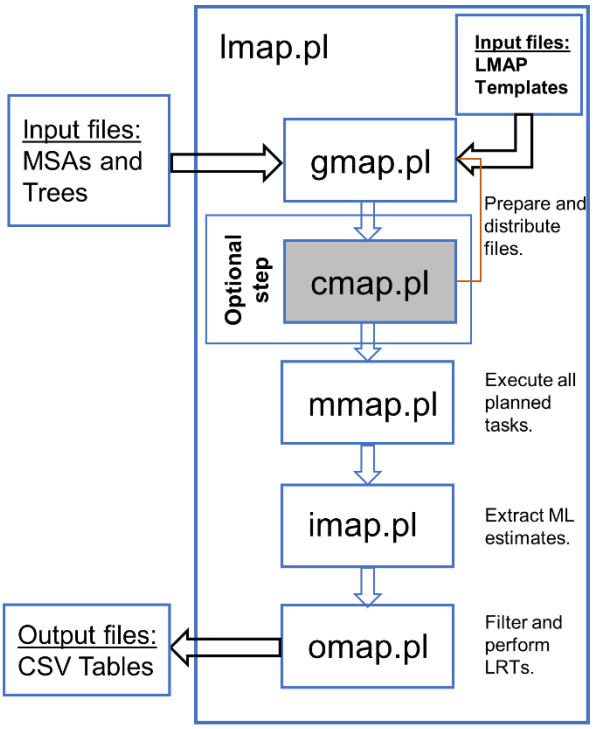


***Figure 10**. lmap.pl application workflow in light of the other LMAP applications.*

The options required to perform the workflow are essentially the same options from gmap.pl. The options -A and -T for input data, the location of the project (option -d), the project name (-j) and the MODELTYPES required to be executed and extracted (option -m). Additionally, extra options are available for any adjustment to different scenarios.

This application limits the functionalities offered from the other applications to those which are required to execute the workflow with no interruptions. This means that functionalities such as, option -t from gmap.pl, or option -x from mmap.pl are not available.

Examples:

```
$ lmap.pl -A Data/MSA -T Data/Trees -d . -j MyDirectoryStruct -m b,w,s[0:1:2:3:7:8:8a],c -n 30
```

```
$ lmap.pl -A Data/MSA -T Data/Trees -d . -j MyDirectoryStruct -m b,w,s[0:1:2:3:7:8:8a],c -n 30 -O
b[0.1,0.5,0.6]:c[0.25,0.35,0.15] -K s[0.1,0.5,0.6]:w[0.25,0.35,0.15]
```

```
$ lmap.pl -A Data/MSA -T Data/Trees -d . -j MyDirectoryStruct -m b,w,s[0:1:2:3:7:8:8a],c -n 30 -e myemail@univ.fac.com
```

```
$ lmap.pl -A Data/MSA -T Data/Trees -d . -j MyDirectoryStruct -m b,w,s[0:1:2:3:7:8:8a],c -n 30 -g f:icode -L 4
```

```
$ lmap.pl -A Data/MSA -T Data/Trees -d . -j MyDirectoryStruct -m b,w,s[0:1:2:3:7:8:8a],c -n 30 --no-omap
```

In all examples the option -n (from mmap.pl) defines the number of *codeml* tasks that will be executing and consequently occupying the same number of CPU cores. The options -O and -K (from gmap.pl) define the initial omega and kappa values for all MODELTYPES (see section **3.1.1**). The option -e triggers a notification to be sent (see section **5.3**). Both options -g and -L (from cmap.pl) must be defined in order to enable the cmap.pl functionality. On the other hand, the option --no-omap, will disable omap.pl functionality in the workflow, thus ensuring that imap.pl is executed last.

An advantage of using this application comes from the fact that imap.pl is not prepared to retrieve results for all MODELTYPES in one run. The lmap.pl automatically generates all results for all MODELTYPES that were indicated in option -m, in separate files.
In this sense, the lmap.pl execution will choose one of the generated files to be loaded in omap.pl. The remaining files will be loaded by the user as per his/her requirements.

8.1. LMAP Over time

An additional feature of lmap.pl, is related to how it handles executions of different MODELTYPES over time for the same project/dataset, when some part of it was already prepared (see section **3.1**). By saving information of which models are executed for a project, lmap.pl implements a functionality that avoids the re-execution of MODELTYPES that have been previously run in another time. The only requirement is the selection of the required models with option -m. Essentially, those MODELTYPES that have been executed will not be rescheduled unless specified by the user through this option. For instance, assuming all the BRM have been identified in the input dataset, but he/she may decide to run, in a first execution, BM (option -m b) and later (same/other day) in a second execution, BSM (option -m w). This will mean that in the second execution, the first MODELTYPES will not be rescheduled (unless if so required in option -m). Even if in case the second execution had been performed in a time previous to the first, then it will not pose a problem, since the user is requiring the (re)execution of the same BSM, but not of the other BM. To conclude, only the models indicated in lmap.pl option -m, will be executed, whereas the remainder (if any were executed) will not be rescheduled. This feature will work in any situation with any methods and whenever using lmap.pl application.

9. HELP OF LMAP APPLICATIONS

9.1. SYNOPSIS Section

The SYNOPSYS sections in applications help (displayed by using the command-line option -h), have the purpose to quickly elucidate the user on how to write the command for the application at hand by giving the correct format for each command-line option and argument (if any).

```
NAME:

  gmap.pl - Generate an organized directory structure.

SYNOPSIS:

  gmap.pl -A [dirfiles] -T [dirfiles] -d [location] -m [s[models],b,w,c] {-j [projname]} {-O b[x,y,...,z]} {-K s[x,y,...,z]}
  gmap.pl -A [dirfiles] -t [inittree.nwk] -d [location] -m [s[models],b,w,c] {-j [projname]} {-O c[x,y,...,z]} {-K w[x,y,...,z]}
```

```
NAME:

  cmap.pl - Change PAML control files specific parameters.

SYNOPSIS:

  cmap.pl -d [directory] -f [e:ctl] -g f:param -L "value" {-r}
```

```
NAME:

  mmap.pl - Run multiple PAML instances.

SYNOPSIS:

  mmap.pl -d [dirpath] {-r} {-n [int]} {-f [e:ctl]} {-p [l:progname]} {-R [resfilename]} {-l} {-e {emailaddr}}
```

```
NAME:

  imap.pl - Extract PAML information from results files.

SYNOPSIS:

  imap.pl -d [dirstruct] [-s [models] | -b | -w | -c] -o [outfile.csv] {-R [res.filename]} {-r}
```

```
NAME:

  omap.pl - Organize CSV files from imap.pl and perform LRT tests.

SYNOPSIS:

  omap.pl -i [infile.csv] -o [outfile.csv]
```

```
NAME:

  lmap.pl - Lightweight Multigene Analyses in PAML.

SYNOPSIS:

  lmap.pl -A [dirfiles] -T [dirfiles] -d [location] -m [s[models],b,w,c] -j [projname]
          {-f [e:ctl]} {-p [l:codeml]} {-R [resfilename]} {-n [integer]} {-e {emailaddr}} {-O [b[x,y,...,z]]} {-K [s[x,y,...,z]]}
```

As shown in the above figures, the mandatory options are presented the argument description enclosed in brackets (e.g.: “-j [projname]”). Otherwise non-mandatory options are found enclosed in curly braces (e.g.: “{-n [integer]}”. Overall, curly braces have the meaning of “optional”, while the brackets, the meaning of mandatory “argument position”.

Whenever an option is specified, an argument (value) must follow, with the exception being the option -e which allows the argument to be missing. The SYNOPSIS in imap.pl has the additional character “|” which means alternative options, hereby requiring that only one option from the alternatives is mandatory, which itself may also require specific arguments. These meanings are also extended to the commands in interactive help menus from gmap.pl and omap.pl.

10. REFERENCES

1. Stajich JE, Block D, Boulez K, Brenner SE, Chervitz SA, Dagdigian C et al. The Bioperl toolkit: Perl modules for the life sciences. Genome research. 2002;12(10):1611-8. doi:10.1101/gr.361602.

2. Yang Z. PAML 4: phylogenetic analysis by maximum likelihood. Molecular biology and evolution. 2007;24(8):1586-91. doi:10.1093/molbev/msm088.

3. Weadick CJ, Loew ER, Rodd FH, Chang BS. Visual pigment molecular evolution in the Trinidadian pike cichlid (Crenicichla frenata): a less colorful world for neotropical cichlids? Molecular biology and evolution. 2012;29(10):3045-60. doi:10.1093/molbev/mss115.