Hammlet – 4-LIN command line instruction

Requirements:

Hammlet (Hybridization Models Maximum Likelihood Estimator) is working on python version 2.7, 3.6, or older. If python dependencies are not installed automatically, the following standard python packages must be preinstalled: SciPy, click, tabulate, colorama (e.g., `python -m pip install scipy click tabulate colorama`).

For installation of hammlet on UNIX-based systems the user can download the source files from GitHub and use pip for installation:

```
git clone https://github.com/ctlab/hammlet
https://github.com/RetroWWU/4-LIN
cd hammlet
pip install .
```

For systems not directly supporting python, like Microsoft Windows, and for preventing package version conflicts on Linux-like systems or MacOS, installing the miniconda environment and using the wheeled container downloaded from the 4-LIN server is recommended (see https://docs.conda.io/en/latest/miniconda.html).

For installing hammlet under miniconda, use pip:

e.g.: `pip install Hammlet-1.7.0.post34-py2.py3-none-any.whl`

Usage:

The command line version of hammlet contains:

- global options or commands
- options and arguments.

Examples:

hammlet --version	prints the current version of hammlet
hammlet --help	prints instructions about commands

```
hammlet draw -m H1 -T1 0.5 -T3 1.0 -g1 0 -g3 1 -o example.svg --names A,B,C,D
```

- building tree T2 with the order of species ABCD and saving the picture in svg format
- command: draw
- options: -m -T1 -T3 -g1 -g3 -o –names
- arguments: H1, 0.5, 1.0, 1, 0, [A,B,C,D]
- out file: example.svg

Commands:

mle-nr	- performs maximum likelihood for selected models
bootstrap	- bootstrap of the maximum likelihood for the selected model

bootstrap-ll	- bootstrap of the maximum likelihood ratio for two selected models from two different groups
calculate-aij	- restoring set of $a_{i,j}$ from parameters of selected tree. Set of Poisson-distributed random variables $a_{i,j}$ reflecting the best set of markers ($y_{i,j}$) for the tree chosen
draw	- drawing tree/hybridization network picture from parameters of the tree, creating graphical representation of a tree
stat-levels	- perform automatic statistical assessment of the user of markers by *stepwise* method
stat-reverse	- perform an automatic statistical assessment of the user of markers by *reverse* method

Local help system

Example: hammlet mle-nr -h (specific help for command: *mle-nr*).

Shared options: Command options mostly share functions and arguments, except *draw*, which has an independent option system.

Used data setup:

-y (y11 y12 y13 y14 y22 y23 y24 y33 y34 y44) = an option is requesting user-defined arguments ordered by a space-delimited set of ten integer values $y_{i,j}$; usage in *mle-nr*, *bootstrap*, *stat-levels*, *stat-reverse*.

-r (r1 r2 r3 r4) – additional option allows the manipulations of branch lengths reflecting a proportional model of effective population sizes on the tree branches; requesting arguments in a space-delimited list of four positive values r_i; default values [1 1 1 1]; common usage between *mle-nr*, *bootstrap*, *bootstrap-ll*, *calculate-aij* , *stat-levels*, *stat-reverse*.

-m (model name) - required option in *mle-nr*, *bootstrap*, *bootstrap-ll*, *calculate-aij* where the user gives the name of the model; this option is extended in *mle-nr* where the user could use not only individual models but select specific sets of models by aliases: All, H1, H2 or by a comma-delimited list of model names. In *draw*, this option uses only H1 or H2 as arguments.

-x [*--exclude*] (model name(s)) - excluding one or several models from the set of models; common usage between *mle-nr*, *bootstrap-ll*, *stat-levels*, *stat-reverse*.

--permutation (1234) fixing the permutation for model(s); permutation should be given as four-digit number using only 1,2,3, and 4 digits, order of digits defining permutation; common for *bootstrap* and *mle-nr*; however, in *mle-nr* syntax of option is --only-permutation.

-t [*--theta*] (n0, T1, T3, g1, g3) require five space delimited values of tree parameters in order: n_0, T_1, T_3, γ_1, and γ_3 for calculating ten values for $a_{i,j}$; common usage between *calculate-aij, bootstrap-ll.* **Note:** $a_{i,j}$ reflecting ideal values for user-defined values $y_{i,j}$ according to the given tree. In contrast to obligatory integer values for $y_{i,j}$, values for $a_{i,j}$ are not negative, and for replacing $y_{i,j}$ by values of $a_{i,j}$ last values must be rounded. As a result, the parameters of an optimized tree after rounding could be slightly different from the initial tree parameters.

Optimization tuning (optional):

--*method* (SLSQP|L-BFGS-B|TNC) selecting an optimization method where optimization is necessary (default method is SLSQP); common usage in *mle-nr*, *bootstrap*, *bootstrap-ll*, *stat-levels*, and *stat-reverse.*

--*theta0* (n0, T1, T3, g1, g3) space-separated list of five initial values of tree parameters required when optimization is started (default values are 100, 1, 1, 0.5, 0.5); common usage in *mle-nr*, *bootstrap*, *bootstrap-ll*, *stat-levels*, and *stat-reverse.*

Debug:

--*debug* (no options) showing all suppressed information about calculations and optimization results and warnings; common usage between *mle-nr*, *bootstrap*, *bootstrap-ll*, *calculate-aij* , *stat-levels*, and *stat-reverse*.

Bootstrap options:

-*n* [--*times*] (number of repetitions) required in *bootstrap*, *bootstrap-ll* or optional in *stat-levels*, *stat-reverse* (here default value is 1000); user-defined integer value for numbers of repeats in bootstrap procedures or the generation of Empirical distributions in statistic commands.

--*output-bootstrap* (path, file_name) defining output for calculated results of Poisson-based randomization and subsequent optimization; usage is common for *bootstrap* and *bootstrap-ll*. However, output formats differ between commands (see below: Output formats).

Statistical options:

-*p* [--*pvalue*] (p-value) option to select critical p-values. Default value is 0.05; range is from 1 to 0. The parameter is common between *stat-levels* and *stat-reverse*. **Note:** because a high p-value by definition may cause Type I errors in statistical comparisons, the value should be equal to or below 0.05. In contrast, a p-value less than 0.01 could underestimate weakly supported phylogenies.

--*ecdf* (no arguments); the option is switching statistical criterion selection between default *chi-square* and *empirical distribution*; common between *stat-levels* and *stat-reverse*. This option requires no specific arguments but has several options used and working only with --*ecdf* switch: -*n* [--*times*] (size of sample) setting size of the empirical sample (default: 1000); --*use-best-senior-model* (no arguments) changing generation of empirical sample from “level against model” mode to “model against model” mode.

--*ecdfs* (4 comma-separated critical values) allow the user to apply specific critical values for running statistical tests. The -*p* option is omitted, and no simulations are running if --*ecdf* option is selected. User-specific critical values should be taken according to the p-value and degrees of freedom used in specific statistical algorithms. The parameter is common between *stat-levels* and *stat-reverse*. **Note:** for the stepwise algorithm (*stat-levels*), all four values have 1 degree of freedom, but for the reverse algorithm (*stat-reverse*) number of degrees of freedom is reduced in descending order from 4 to 1; e.g., in the first case, chi-square values for p=0.05 will be taken as 3.84, 3.84, 3.84, 3,84; and in second case these values should be taken as 9.49, 7.82, 5.99, 3.84.

--output-mle (path, file_name) giving way for saving maximum likelihood optimization results for all taken models in statistical algorithms; common between *stat-levels*, *stat-reverse*, and *mle-nr* (see below: Output formats).

--output-result (path, file_name) giving way for communication with 4-LIN R interface; common between *stat-levels* and *stat-reverse*. The resulting file contains information about the parameters of the selected essential tree and some additional values for tuning the 4-LIN interface screen (see below: Output formats).

Specific options:

***mle-nr*:**

--*best* (number of models, *all*) is restricting MLE output to a selected number of results starting from the highest likelihood value.

--*only-first-permutation* (no arguments) restricts the MLE optimization to first permutation (1234) for all modes.

--*only-non-redundant-permutations* (no arguments) showing all permutations for all models. **Note:** in *mle-nr*, only non-redundant sets of modes and tees (permutations) exist, excluding all phylogenetically identical models and symmetrical permutations. The option is switching off the selection of permutations for the non-redundant set and showing all permutations, but not identical models. The option --*only-permutation* (described earlier) working in a not restricted area without selection of options --*only-non-redundant-permutations*.

--*no-polytomy* (no arguments) suppresses the output of polytomy MLE results.

***bootstrap-ll*:**

-l [-level] (level) is setting up the level for "level against model" bootstrap mode. The level should be given in "N," and the number of related groups (e.g., group ϴ4 should be converted to level N4). **Note:** To use "model against model" bootstrap mode, it is necessary to set up a level and exclude all models from this level except one; e.g., in group ϴ4, two models are defined: 2H1 and 2H2, then excluding model 2H2 by option "-x 2H2" setting up "model against model" bootstrap with senior model 2H1.

***calculate-aij*:**

--*output-aij* (path, file_name) giving the way for output of twelve comma-delimited $a_{i,j}$ values.

draw:

-o [--output] (path, file_name) setting the way for saving a resulting picture of

tree/network. **Warning**! Command *draw* creates an image in vector (svg) format, and for correct recognition of the resulting picture file by system programs in non-UNIX systems user must set .svg extension to the output file name. On UNIX or Linux systems, the user could apply, e.g., rsvg command from librsvg2-bin for conversion of svg format to other formats.

-T1 (value) value of parameter T_1 for a selected tree.

-T3 (value) value of parameter T_3 for a selected tree.

-g1 (value) value of parameter γ_1 for a selected tree.

-g3 (value) value of parameter γ_3 for a selected tree.

-n [--names] (four names) four comma-separated names of species; (default: 1,2,3,4). **Note:** Names of species should not contain spaces. We recommend using underlines. And *-n* [*--names*] in <u>*draw*</u> should not be mixed with *-n* [*--times*] in other commands.

-a, [*--width*] (value) width of resulting picture resolution in points (default: 600).

-b, [*--height*] (value) height of resulting picture resolution in points (default: 400).

--threshold-T (value) setting up threshold for almost-zero T (default: 0.01).

--threshold-g (value) setting up threshold for almost-zero gamma (default: 0.01).

-cb, [*--color-background*] (color) setting up background color (default: transparent).

Note: The user can set colors in verbose format (e.g., black) or in hexadecimal form (e.g., FFFF).

-ct [*--color-tree*] (color) Tree branches color (default: blue).

-ch [*--color-hybrid*] (color) Hybridization lines color (default: red).

-cr [*--color-ruler*] (color) Ruler color (default: green).

--ruler [*--no-ruler*] draw a ruler equivalent to 1 for tree branch length (default: ruler on). **Note:** to switch off a ruler set *--no-ruler* switch.

Output formats:

--output-mle (<u>*mle-nr*</u>, <u>*stat-levels*</u>, and <u>*stat-reverse*</u>)

The option provides the optimal results for a selected list of models within the user-defined file. The output file contains a comma-delimited table with nine columns. The first row contains header values where:

1. <u>Model</u> reflects the <u>name</u> of the model

2. <u>Mnemo</u> reflects <u>alias</u> in TTgg nomenclature where, e.g., in H1:TTg1 first two letters indicate the root of the model (in this case: 2H1), and the last four letters reflect conditions of model parameters (in this case TT means that T_1 and T_3 are variable, and g1 indicate that γ_1 is variable, but γ_3 is a fixed equivalent to 1)

3. <u>Perm</u> reflects permutation according to specific models. 1234 means that we retain the initial order of species

4. <u>LL</u> reflects maximal logarithmic likelihood values found by optimization for a given model and permutation;

5. - 9. <u>n0</u>, <u>T1</u>, <u>T3</u>, <u>g1</u>, and <u>g3</u> reflect five model parameters (n_0, T_1, T_3, γ_1, and γ_3) derived during the optimization procedure.

--output-bootstrap (<u>*bootstrap*</u>)

The option provides results for optimization of the selected model, permutation, and $y_{i,j}$ values under the Poisson-based randomization algorithm of $y_{i,j}$ values. The output file contains a comma-delimited table with 16 columns (in the first row, we have seven columns) where:

1.-10. y reflects result of randomization by Poisson-based algorithm of user-specified $y_{i,j}$ values in order: $y_{1,1}$; $y_{1,2}$; $y_{1,3}$; $y_{1,4}$; $y_{2,2}$; $y_{2,3}$; $y_{2,4}$; $y_{3,3}$; $y_{3,4}$; $y_{4,4}$. However, the real order of applied $y_{i,j}$ in the optimization process depends on the --*permutation* option. The report given in order reflects the initial permutation (1234).

11. LL reflects the maximal logarithmic likelihood value found by optimization for the current dataset. We have a fixed model, permutation, and randomized set of $y_{i,j}$ values in the dataset.
12.-16 n0, T1, T3, g1, and g3 reflect the optimization procedure's five model parameters (n_0, T_1, T_3 γ_1, and γ_3).

--*output-bootstrap* (*bootstrap-ll*)

The option provides results for the log-likelihood ratio between the best permutation of the model from a selected group and the best permutation of the simpler model. The output file contains a comma-delimited table with 17 columns (in the first row, we have eight columns) where:

1.-10. y reflects result of randomization by Poisson-based algorithm calculated from given model parameters $y_{i,j}$ values in order: $y_{1,1}$; $y_{1,2}$; $y_{1,3}$; $y_{1,4}$; $y_{2,2}$; $y_{2,3}$; $y_{2,4}$; $y_{3,3}$; $y_{3,4}$; $y_{4,4}$.

11. Mx reflects the name of the complex model, having the highest maximum likelihood between all models in a selected group.

12. px reflects permutation for this complex model.

13. LLx reflects the log-likelihood value for this complex model.

14. My reflects the name of the simpler model (stable: fixed by user).

15. py reflects permutation for this simpler model (Note: initial permutation for this model is the first permutation (1234) because the conversion of model parameters to $y_{i,j}$ values generate first permutation (1234)).

16. LLy reflects the log-likelihood value for this simpler model.

17. LLx-LLy reflects the log-likelihood ratio for complex and more straightforward models.

--output-result (*stat-levels* and *stat-reverse*)

Option providing string for communication between hammlet and R-based 4-LIN interface. The report contains comma delimited string where the user could find in order: statistical method; model group (level); model name; model alias; model permutation; log-likelihood value; n_0; T_1; T_3; γ_1; γ_3; the other three values are technical and not interest for users.

Consol output:

A difference between console and output streams is a tab-delimited on the console and a comma delimited in the reports. For *mle-nr*, *bootstrap*, and *bootstrap-ll* standard output provides the same table format as described in the respective sections of the output formats chapter. For *calculate-aij* results of calculations show ten $a_{i,j}$ values on the screen. In contrast, for statistical algorithms, *stat-level* and *stat-reverse* console output provide comprehensive information about comparisons.

A significant output of statistical algorithms started from the resulting table (after signal [+] MLE results (best per level)). The table consists of 10 columns where:

1. Level reflects model group (level)
2. Model reflects the name of a model
3. Mnemo reflects the alias of a model in TTgg nomenclature
4. Perm reflects permutation according to a model
5. LL reflects log-likelihood values for model and permutation
6. - 10. n0, T1, T3, g1, and g3 reflect the optimization procedure's five model parameters (n0, T1, T3, g1, and g3).

The order of models in the table depends on the selected algorithm. In the stepwise algorithm (***stat-levels***) order of model groups starts from group Ө4 and descend to group Ө0; in the reverse algorithm (***stat-reverse***), this order also starts from group Ө4, but next ascends from group Ө0 to group Ө3.

After the table with the optimization results, the user can see the process of searching for an essential tree, which includes several steps (minimum one step). All steps are marked by the following symbol: "[*]". Next, the user can see what groups are compared in the current step (e.g., N4-N1 mean comparing two best models from groups Ө4 and Ө1). Next, information depends on the selected statistical criterion: chi-square or eCDF (empirical distribution). The chi-square-based algorithm shows the degrees of freedom (df =), a value of duplicated log-likelihood ratios (stat =), and respected p-values (p-value =). For the eCDF criterion, the user can see values of duplicated log-likelihood ratios (2*delta LL =), critical values calculated by the empirical distribution algorithm (critical LL =), and details of critical value selection (in parenthesis). For user-defined critical values, (--ecdfs switch) output string is similar to eCDF string without details of critical value selection.

After selecting the resulting essential tree, two "[+]" signals are shown with used conditions for selection. The next value indicates the chosen group and model ("Final result: level [level], model [model]").

Note: hammlet output is limited due to the text mode. However, the user could create a picture of the resulting tree by the *draw* command where necessary parameters of the tree, model, and permutation could be extracted from the resulting table, taking into account the "Final result:…" string. From the field Mnemo, the user can get arguments for draw option -m (--model) (H1 or H2 are the first two symbols in this field); arguments for draw options *-T1*, *-T3*, *-g1*, *-g3* user could get from respected fields of the same table.

Examples:

For some examples, we will use a synthetic distribution of ten presence/absence patterns giving results at the significance level 0.05 as 1H2 and 0.01 as 1HP1 with the *reverse* algorithm. The ten $y_{i,j}$ values are: 8 17 6 26 3 3 12 3 1 2. Parameter set for 1H2 is (n_0=42.428, T1=0.422, T3=0.420, g1=1, g3=0.829), permutation is 4123; and for 1HP1 set is (n0=43.169, T1=0, T3=0.726, g1=0 (not defined), g3=0.367), and permutation is 1243.

mle-nr:

We are optimizing all available models and permutations for non-redundant sets of models:

```
hammlet mle-nr -y 8 17 6 26 3 3 12 3 1 2 -m All --output-mle mle-example1.out
```

Now we are restricting results to the ten best models and fixing permutation to 1234.

```
hammlet mle-nr -y 8 17 6 26 3 3 12 3 1 2 -m All --best 10 --only-first-permutation --output-mle mle-example2.out
```

bootstrap:

We are running bootstrap (Poisson-based randomization) of our data on model 1H2 and permutation 4123 one hundred times.

```
hammlet bootstrap -y 8 17 6 26 3 3 12 3 1 2 -m 1H2 --permutation 4123 -n 100 --output-bootstrap bootstrap-example1.csv
```

bootstrap-ll:

Here we are running the bootstrap of the likelihood ratio for group ϴ4 against model 1H2 from group ϴ3 with known parameters one hundred times.

```
hammlet bootstrap-ll -l N4 -m 1H2 -t 102.5 0.44 0.42 1 0.83 -n 100 --output-bootstrap bootstrap-example2.csv
```

We will run the “model-against-model” bootstrap with the same data but excluding model 2H2 from level N4.

```
hammlet bootstrap-ll -l N4 -x 2H2 -m 1H2 -t 102.5 0.44 0.42 1 0.83 -n 100 --output-bootstrap bootstrap-example3.csv
```

calculate-aij:

We will get values $a_{i,j}$ from parameters of model 1H2 (for fixed permutation 1234).

```
hammlet calculate-aij -m 1H2 --theta 102.5 0.44 0.42 1 0.83 --output-aij aij-example.csv
```

stat-levels:

We will perform statistical tests on our data, according to the *stepwise* algorithm and chi-square criterion, at the level of significance 0.05. Additionally, we will get MLE results in the respected file (note: model T0 will be excluded from the next calculations to get compatibility with 4-LIN, where T0 is excluded permanently (T0 is “shadow” of model T1 with equaling branch length and could not be correctly compared with the T1 normal way)).

```
hammlet stat-levels -y 8 17 6 26 3 3 12 3 1 2 --exclude T0 -p 0.05 --output-mle stat-mle-examle.csv
```

Now we will repeat the same test but with eCDF criterion. The output we will get for results (Note: running not for seconds, but tens of minutes).

```
hammlet stat-levels -y 8 17 6 26 3 3 12 3 1 2 --exclude T0 --ecdf -n 500 --output-result stat-level-ecdf-examle.csv
```

We will use the same data, and instead of the chi-square criterion will use the mixture with 0 point mass and chi-square with one degree of freedom for comparing level N1 with polytomy. Other comparisons will use the classical chi-square criterion. And the last example will be about the --ecdfs option. Here will be only a console output.

```
hammlet stat-levels -y 8 17 6 26 3 3 12 3 1 2 --exclude T0 --ecdfs 3.84,3.84,3.84,1.92
```

stat-reverse:

Here we will use the same three examples but with the reverse algorithm.

We will perform statistical tests on our data, with chi-square criterion at the level of significance 0.05. Additionally, we will get MLE results in respected files.

```
hammlet stat-reverse -y 8 17 6 26 3 3 12 3 1 2 --exclude T0 -p 0.05 --output-mle stat-mle-examle2.csv
```

Next, we will repeat the same test but with eCDF criterion. The output we will get for results.

```
hammlet stat-reverse -y 8 17 6 26 3 3 12 3 1 2 --exclude T0 --ecdf -n 500 --output-result stat-reverse-ecdf-examle1.csv
```

Next, we will apply the "model-against-model" mode for the generation of an empirical sample (experimental):

```
hammlet stat-reverse -y 8 17 6 26 3 3 12 3 1 2 --exclude T0 --ecdf -n 500 --use-best-senior-model --output-result stat-reverse-ecdf-examle2.csv
```

For comparison with polytomy, we will use a complex chi-square distribution mixture with binary coefficients where df = 4 to calculate necessary critical values. Last, we will use the --ecdfs option, considering descending degrees of freedom from 4 to 1. There will be only a console output.

```
hammlet stat-reverse -y 8 17 6 26 3 3 12 3 1 2 --exclude T0 --ecdfs 6.50,7.82,5.99,3.84
```

draw:

Now we will make pictures of our model 1H2 with ruler and model 1HP1 without with respected permutations (order of species).

For 1H2, we know the parameters and the permutation. We will call our species A, B, C, and D. We must fit it to a known permutation to get the correct species order on the tree (4123).

```
hammlet draw -m H1 -T1 0.44 -T3 0.42 -g1 1 -g3 0.83 -n D,A,B,C -o 1H2-example.svg
```

Now we will repeat all steps for parameters of model 1HP1 and permutation 1243, but without a ruler.

```
hammlet draw -m H1 -T1 0 -T3 0.726 -g1 0 -g3 0.369 -n A,B,D,C --no-ruler -o 1HP1-example.svg
```

With the given examples, the user can replace command-line options arguments with their data and settings.