

TABLE OF CONTENTS

- ▶ [Home.md](#)
- [I.-Quickstart.md](#)
- [II.-Installation.md](#)
- [III.-Usage-guide.md](#)
- [IV.-Output-files.md](#)
- [V.-Optimising-input-parameters.md](#)
- [VI.-Prior-clusters.md](#)

Home.md

PhyCLIP (*Phylogenetic Clustering by Linear Integer Programming*)

PhyCLIP is an integer linear programming (ILP) approach that optimally delineates a tree into statistically-principled clusters. Other than a **rooted** phylogeny, 3 additional inputs are required from the user:

1. Minimum number of sequences (S) that can be quantified as a cluster.
2. Multiple of deviations (γ) from the grand median of the mean pairwise sequence patristic distance that defines the within-cluster divergence limit.
3. False discovery rate (FDR) to infer that the diversity observed for every combinatorial pair of output clusters is significantly distinct from one another.

A manuscript describing PhyCLIP is available here:

MANUSCRIPT LINK

We highly encourage that you go through this documentation before starting your analysis. However, if you would like to promptly run PhyCLIP using the recommended procedure, go to [Quickstart](#).

Contents:

- I. [Quickstart](#)
- II. [Installation](#)
- III. [Usage guide](#)
- IV. [Output files](#)
- V. [Optimising input parameters](#)
- VI. [Prior clusters](#)

I.-Quickstart.md

N.B. We highly encourage that you go through all of the documentation before starting your analysis.

Quickstart guide (for academic users, Python 2.7 support only)

1. **Installation**
2. Install dependencies using [Anaconda](#).

```
$ conda install -c etetoolkit ete3
$ conda install numpy scipy statsmodels
```

3. Install Gurobi (linear programming solver) using Anaconda as well.

```
$ conda config --add channels http://conda.anaconda.org/gurobi
$ conda install gurobi
```

4. Obtain Gurobi license.

Method I: If your machine is connected to the internet via a recognized academic domain (e.g. '.edu' addresss)

- Register a FREE account via <http://www.gurobi.com/registration/academic-license-reg>
- Login and access <http://user.gurobi.com/download/licenses/free-academic>
- Follow instructions on page to retrieve your license or go to [Installation](#) for more details.

Method II: If your machine is **NOT** connected via an academic domain but you can verify that you are an academic user.

- To be added.

5. Download and install PhyCLIP

```
$ cd /path/to/PhyCLIP-master/
$ python setup install
```

6. Input files

7. Phylogenetic tree in **NEWICK** format.

- Tip: Root your phylogenetic tree with an appropriate outgroup. If that is not possible, then by mid-point rooting. You can do this easily with [Figtree](#).

8. Prepare the following PhyCLIP input text file to run various ranges of input parameters (see [Optimising-input-parameters](#) for details).

```
/path/to/input_newick_tree.nwk
3-10(1),0.05-0.20(0.05),1-3(0.5)
```

9. Run PhyCLIP

10. For intermediate-resolution clustering (i.e. broadly defined clusters that are still well-separated but encompasses the majority of data in the tree):

```
$ phyclip.py --input_file </path/to/input_file.txt> --collapse_zero_branch_lengths 1 --pdf_tree
--optimise intermediate
```

11. For high-resolution clustering (i.e. phylogenetic tree delineated into a large number of small, high confidence clusters with low average internal divergence, tolerating a higher number of unclustered sequences):

```
$ phyclip.py --input_file </path/to/input_file.txt> --collapse_zero_branch_lengths 1 --pdf_tree
--optimise high
```

N.B. The command in either case above will collapse internal nodes with zero branch lengths. See [Usage-guide](#) to understand why this is recommended.

4. Analyse results

5. Open **pdftree_optimal_parameter_{S}{FDR}{gamma}_{tree_filename}.pdf** for quick check of the clustering output with the optimal set of input parameter.

6. Alternatively, if you are using [Figtree](#), you can also check the output annotated NEXUS tree file **tree_optimal_parameter_{S}{FDR}{gamma}_{tree_filename}.tre**

7. A text file listing your sequences and their respective cluster-ID can be found in **cluster_optimal_parameter_{S}{FDR}{gamma}_{tree_filename}.txt**

8. If the clustering result with the optimal set of input parameter is unsatisfactory, you can examine the **summary-**

`stats_{input_text_filename}.txt` file to determine which parameter set ran generated a more ideal clustering result for you (see [Output files](#) and [Optimising-input-parameters](#) for more details).

II.-Installation.md

PhyCLIP is written in Python 2.7 (no support for Python 3 currently) and depends on several python libraries and at least one ILP solver.

To simplify the installation process, we highly recommend that you use Anaconda, a free and open-source distribution of Python and package management system. Visit <http://www.anaconda.com/download/> to download and install the **Python 2.7 version** distribution for your preferred OS.

Prerequisite 1: Python libraries

PhyCLIP depends on several Python libraries:

- numpy, scipy, statsmodels (mathematical/statistical operations)
- ete3 (parsing phylogenetic trees)

To install the dependencies, go to Terminal (Mac/Linux) or Command/Anaconda Prompt (Windows):

```
$ conda install -c etetoolkit ete3
$ conda install numpy scipy statsmodels
```

Alternatively, if you are using PyPi:

```
$ pip install ete3
$ pip install numpy scipy statsmodels
```

Prerequisite 2: ILP solver

PhyCLIP currently supports two ILP solvers. You can choose either **ONE** to install depending on your access to these solvers:

1. **Gurobi** optimizer (<http://www.gurobi.com/>) is a commercial linear and quadratic programming solver with FREE licenses available for academic users.
2. **GLPK** (GNU Linear Programming Kit, <http://www.gnu.org/software/glpk/>) is a free and open-source package intended for solving large-scale linear programming, mixed integer programming, and other related problems.

If you are a university user (i.e. you have internet access from a recognized academic domain, e.g. '.edu' addresss), **we highly recommend running PhyCLIP with the Gurobi solver**. GLPK performs poorly in terms of both speed and solvability (GLPK version 4.65 solved only 2 of the 87 standard test-set mixed-integer programming models whereas Gurobi is the fastest solver for all 87 benchmark problems, see <http://plato.asu.edu/ftp/milpc.html>).

Furthermore, as with any other linear programming problems, it is possible to obtain multiple optimal solutions. Currently, GLPK can only return ONE solution that is guaranteed to be the global optimal if and only if the feasible region is convex and bounded. However, this may not always be the case. Gurobi, on the other hand, generates a solution pool which may include > 1 optimal solution.

Gurobi

The easiest way to install Gurobi is via the Anaconda platform:

1. Make sure you have Anaconda for Python 2.7 installed (see above).
2. Install the Gurobi package via conda:

```
$ conda config --add channels http://conda.anaconda.org/gurobi
$ conda install gurobi
```

3. You need to install a Gurobi licence next. Visit <http://www.gurobi.com/registration/academic-license-reg> to register for a free Gurobi account. Follow the instructions in the verification email from Gurobi to set your password and login to your Gurobi

account via <http://www.gurobi.com/login>.

4. You can now access <http://user.gurobi.com/download/licenses/free-academic> to request for a free academic license. Once requested, you will be brought to the License Detail webpage.
5. To install the license, go to Terminal/Command Prompt: `$ grbgetkey XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX` where `XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX` is your unique license key shown in the License Detail webpage. Note that an active internet connection from a recognized academic domain (e.g. '.edu' addresss) is required for this step.

GLPK

You can easily install GLPK via Anaconda as well:

```
$ conda install -c conda-forge glpk
```

Alternatively, you can also install GLPK from source, go to <http://ftp.gnu.org/gnu/glpk/> and download the latest distribution of glpk as a tarball. You can find installation information in the documentation provided.

Install PhyCLIP

Finally, install phyclip.py by:

```
$ cd PhyCLIP-master/  
$ python setup.py install
```

You may need sudo privileges for system-wide installation. Otherwise, it is also possible to use phyclip.py locally by adding the phyclip_modules folder to your \$PYTHONPATH.

III.-Usage-guide.md

PhyCLIP requires the user to provide a phylogenetic tree and 3 parameters:

1. Minimum number of sequences (S) that can be quantified as a cluster.
2. False discovery rate (FDR) to infer that the diversity observed for every combinatorial pair of output clusters is significantly distinct from one another.
3. Multiple of deviations (γ) from the grand median of the mean pairwise sequence patristic distance that 3. defines the within-cluster divergence limit.

How to use PhyCLIP?

Step 0: Rooting your phylogenetic tree.

As PhyCLIP incorporates topological information of the tree for clustering, prior to running PhyCLIP, we advise that you **root** your input phylogenetic tree with the appropriate outgroup, or if you are unsure, by its mid-point. You can use programs such as [FigTree](#) for tree rooting. Note that the input phylogenetic tree must be in the **NEWICK** format. You can use Figtree as well for file type conversion.

Step 1: Prepare input file.

Prepare the input text file indicating the path of your phylogenetic tree and the list of parameter sets to run in the following format:

```
/path/to/input_newick_tree.nwk  
S,FDR,gamma  
...  
  
E.g. (input_example.txt in examples/ folder):  
examples/example.nwk  
3,0.2,2  
5,0.2,1
```

You can also run a range of input parameters. This is useful if you would like to obtain the cluster configuration by the optimal input parameters (see [Optimising-input-parameters](#) for details.). For example, if you choose to test $_S = 3-10$ (increasing by 1), $FDR = 0.05-0.20$ (increasing by 0.05) and $\gamma = 1-3$ (increasing by 0.5), generate the following input text file:

```
/path/to/input_newick_tree.nwk
3-10(1),0.05-0.20(0.05),1-3(0.5)
```

Step 2: Running PhyCLIP

Basic command

```
$ phyclip.py --input_file </path/to/input_file.txt>
```

Collapsing nodes with zero branch lengths

As mentioned above, the topology of the tree is incorporated in the clustering construct. As such, internal nodes with zero branch lengths, usually arising from bifurcating trees, may lead to erroneous clustering. For example:

PhyCLIP allows the user to collapse all internal nodes with zero branch lengths prior to clustering. By default, any internal node with branch length $\leq 1e-7$ is rounded to zero. You can do so by flagging:

```
$ phyclip --input_file </path/to/input_file.txt> --collapse_zero_branch_lengths {0,1}
```

You may modify the maximum patristic distance to be rounded to zero by using the `--equivalent_zero_length <float>` flag.

Subsuming sub-clusters

There may be cases where PhyCLIP is sensitive to clustering clades that minimally satisfies the minimum cluster size threshold ($_S$). Note that these supposedly over-delineated clusters are **NOT** erroneous results but optimal clades that satisfy the same statistical requisites as the others.

Nonetheless, PhyCLIP provides the user the option to subsume such sub-clusters into their parent clade to better facilitate higher level interpretation, subject to if the statistical framework of the original output is maintained.

There are two such options available:

1. Minimum cluster size threshold sensitivity-induced clusters

```
$ phyclip --input_file </path/to/input_file.txt> --subsume_sensitivity_induced_clusters {0,1}
```

PhyCLIP will subsume any sub-clusters <25-th percentile (default) of the output cluster size distribution if the statistical framework of the original output is maintained. You may change the percentile using `--sensitivity_percentile {INT}` flag.

2. ALL sub-clusters

```
$ phyclip --input_file </path/to/input_file.txt> --subsume_subclusters {0,1}
```

PhyCLIP subsumes all sub-clusters into their parent clade if the statistical framework of the original output is maintained.

N.B. If you flag both options above, PhyCLIP will first subsume the sensitivity-induced clusters, reassess the distribution of the clusters, then proceed with subsuming all sub-clusters based on the revised clusters.

Treeinfo file

As data parsing and statistical calculations may take some time, PhyCLIP generates a `{tree_filename}_treeinfo.txt` file by default when analysing a new phylogenetic tree. This file allows you to quicken the analysis of the SAME tree in future runs.

```
$ phyclip --input_file </path/to/input_file.txt> --treeinfo </path/to/{tree_filename}_treeinfo.txt>
```

Depending on the size of your phylogenetic tree, the filesize of `_treeinfo.txt` may be quite large. You can choose **NOT** to generate

the `_treeinfo.txt` file by:

```
$ phyclip --input_file </path/to/input_file.txt> --no_treeinfo
```

Annotated tree file in PDF format

Apart from the standard output files, you can additionally obtain an annotated tree in PDF format by:

```
$ phyclip.py --input_file </path/to/input_file.txt> --pdf_tree
```

See [Output files](#) for more information on the outputs generated by PhyCLIP.

Statistical methods

PhyCLIP currently supports two ways to calculate robust estimator of scale (i.e. deviations) of a distribution: (i) median absolute deviation (MAD) and (ii) Qn measure (see [Rousseeuw & Croux, 1993](#)). **By default, PhyCLIP uses Qn** which is as robust as MAD (i.e. 50% breakdown point) but is calculated solely using the differences between the values in the distribution without needing a location estimate (i.e. no assumptions of centrality). It has also been proven to be statistically more efficient in both Gaussian and non-Gaussian distributions compared to MAD.

On the other hand, PhyCLIP performs hypothesis testing on the pairwise patristic distance distributions between every pairwise combination of clusters to infer their separateness. The current implementation of PhyCLIP either uses the putative Kolmogorov-Smirnov (KS) test or the Kuiper's test. Although both tests are nonparametric, **PhyCLIP uses Kuiper's test by default** as its statistic incorporates both the greatest positive and negative deviations between the two distributions whereas the KS test statistic is defined only by their maximum difference. As a result, the Kuiper's test becomes equally sensitive to differences to the tails as well as the median of the distributions but the KS test works best when the distributions differ mostly at the median.

We recommend that you do **NOT** change the default statistical methods. However, PhyCLIP does allow the user to change them to her preferred choice by:

- Estimator of scale/Deviation:

```
$ phyclip --input_file </path/to/input_file.txt> --gam_method {MAD,Qn(default)}
```

- Hypothesis testing:

```
$ phyclip --input_file </path/to/input_file.txt> --hypo_test {Kuiper(default),KolSmi}
```

IV.-Output-files.md

For **each input parameter set** (`_S_`, `FDR`, `gamma`), PhyCLIP outputs the following files:

1. `cluster_{gam_method}_{hypo_test}_{sol_index}_{_S_}_{FDR}_{gamma}_{tree_filename}.txt` - Tab-delimited file detailing the cluster-ID of every clustered taxon for the input (`_S_`, `FDR`, `gamma`) parameter set denoted in the filename.
2. `tree_{gam_method}_{hypo_test}_{sol_index}_{_S_}_{FDR}_{gamma}_{tree_filename}.tre` - NEXUS tree with [FigTree](#) annotations of clusters for the input (`_S_`, `FDR`, `gamma`) parameter set denoted in the filename.

Additionally, PhyCLIP also outputs following files for **each run**:

- `summary-stats_{input_text_filename}.txt` - Tab-delimited file summarizing the clustering statistics (e.g. % clustered, grand mean pairwise patristic distance of clusters and its dispersion, etc.) of **ALL** parameter sets.
- IF `--optimise` flag is called (see [Optimising input parameters](#)):
 - `cluster_optimal_parameter_{_S_}_{FDR}_{gamma}_{tree_filename}.txt` - same as cluster file above but for the optimal input parameter set.
 - `tree_optimal_parameter_{_S_}_{FDR}_{gamma}_{tree_filename}.tre` - same as annotated NEXUS tree file above but for the optimal input parameter set.

- `pdftree_optimal_parameter_{_S_}_{FDR}_{gamma}_{tree_filename}.pdf` if `--pdf_tree` flag is called (see below)

Annotated tree file in PDF format

You can retrieve an annotated tree file

(`pdftree_{gam_method}_{hypo_test}_{sol_index}_{_S_}_{FDR}_{gamma}_{tree_filename}.pdf`) in PDF format by appending the flag:

```
$ phyclip.py --input_file </path/to/input_file.txt> --pdf_tree
```

cluster_*.txt file

Each line of a **cluster_*.txt** contains the following tab-delimited information over 4 columns (in order):

1. Cluster-ID
2. Sequence name
3. Original (descendant) Cluster-ID subsumed if `--subsume_sensitivity_induced_clusters 1` is flagged (see [Usage guide](#) for more information)
4. Original (descendant) Cluster-ID subsumed if `--subsume_subclusters 1` is flagged (see [Usage guide](#) for more information)

tree_*.txt file

These tree files are in NEXUS format and annotated with cluster-ID and subtree/node-ID information. The file is written to be opened in [FigTree](#).

summary-stats_*.txt file

The **summary-stats_*.txt** file is tab-delimited and contains the summary statistics of the clustering output for every parameter set ran in a single session. In order, the information found in each column is as follows:

1. Phylogenetic tree filename
2. Minimum cluster size (`_S_`)
3. *FDR*
4. *gamma*
5. Hypothesis test method used (Kuiper or Kolmogorov-Smirnov[KS])
6. Estimator of scale (i.e. Deviation) used (Median absolute deviation[MAD] or Qn)
7. FDR-corrected p-values conducted pre- or post- filtering for minimum cluster size (`_S_`)
8. Within-cluster divergence limit (*WCL*)
9. Solution index (if there are multiple solutions, starts from 0)
10. % of prior sequences clustered by PhyCLIP (ONLY if you implemented a prior [see [Prior clusters](#)])
11. Number of clustered sequences
12. Total number of sequences in tree
13. % of sequences clustered by PhyCLIP
14. Total number of clusters
15. Mean output cluster size
16. Standard deviation of output cluster size distribution

17. Median output cluster size
18. Median absolute deviation of cluster size distribution
19. Smallest output cluster size
20. Largest output cluster size
21. Grand mean of mean pairwise sequence patristic distance across all output clusters
22. Standard deviation of mean pairwise sequence patristic distance distribution
23. Grand mean of median pairwise sequence patristic distance across all output clusters
24. Standard deviation of median pairwise sequence patristic distance distribution
25. Lowest mean pairwise sequence patristic distance
26. Highest mean pairwise sequence patristic distance
27. Mean inter-cluster patristic distance
28. Standard deviation of inter-cluster patristic distance distribution
29. Median inter-cluster patristic distance
30. Median absolute deviation of inter-cluster patristic distance distribution
31. Lowest inter-cluster patristic distance
32. Highest inter-cluster patristic distance

V.-Optimising-input-parameters.md

Why optimise input parameters?

PhyCLIP's user-defined parameters ($_S_$, γ , FDR) can be calibrated across a range of input values to optimise the statistical properties of the clustering results so as to select an optimal parameter set and its associated clustering result.

Parameter optimisation criteria should be prioritised by the research question, as the clustering resolution and cluster definition are dependent on the question, and therefore the degree of information required to capture variant ecological, epidemiological and/or evolutionary processes of interest.

How to determine the optimal input parameter set?

For every clustering result of an input parameter set, PhyCLIP calculates the:

1. Grand mean of the pairwise patristic distance distribution (μPWD) and its standard deviation
2. Mean of the inter-cluster distance (μICD) and its standard deviation
3. Percentage of sequences clustered ($_ \% _$)
4. Total number of clusters ($_ TC _$)/Mean cluster size (μCS)

Each of the above statistic can be used to respectively optimised for the following:

1. Clustering configuration with the lowest average internal divergence
2. Clustering configuration with well-separated clusters
3. Minimise number of unclustered sequences
4. Tolerable level of stratification of the tree.

Recommended procedure

We recommend running PhyCLIP using a combination of various input parameter range.

For example, if you choose to test $_S_ = 3-10$ (increasing by 1), $FDR = 0.05-0.20$ (increasing by 0.05) and $\gamma = 1-3$ (increasing by 0.5), generate the following input text file:

```
/path/to/input_newick_tree.nwk
3-10(1),0.05-0.20(0.05),1-3(0.5)
```

Next, type the following command in your Terminal/Command Prompt.

For high-resolution clustering (i.e. phylogenetic tree delineated into a large number of small, high confidence clusters with low average internal divergence, tolerating a higher number of unclustered sequences; in order of priority: min *muPWD*, max *mulCD*, max $_ _ _$):

```
$ phyclip.py --input_file </path/to/input_file.txt> --optimise high
```

For a more intermediate resolution (i.e. broadly defined clusters that are still well-separated but encompasses the majority of data in the tree; in order of priority: max $_ _ _$, min *muPWD*, max *mulCD*):

```
$ phyclip.py --input_file </path/to/input_file.txt> --optimise intermediate
```

After completing all analyses, PhyCLIP will additionally generate the following output files:

- **cluster_optimal_parameter_{_S_}_{FDR}_{gamma}_{tree_filename}.txt** - Tab-delimited file detailing the cluster-ID of every clustered taxon for the optimal input parameter set denoted in the filename.
- **tree_optimal_parameter_{_S_}_{FDR}_{gamma}_{tree_filename}.tre** - NEXUS tree with [FigTree](#) annotations of clusters for the optimal input parameter set denoted in the filename.
- **pdfree_optimal_parameter_{_S_}_{FDR}_{gamma}_{tree_filename}.pdf** if `--pdf_tree` flag is called (see [Output files](#))

Alternatively, if you prefer to optimise with a different set of clustering statistics, you can do so by examining the **summary-stats_{input_text_filename}.txt** (see [Output files](#)) which summarizes these statistics for the range of parameter sets analysed.

VI.-Prior-clusters.md

Running PhyCLIP with prior clustering information

N.B. Prior implementation is **ONLY** supported with the Gurobi ILP solver (see [Installation](#))

If prior information that certain sequences should be clustered together is known, PhyCLIP allows the user to input such prior information into the ILP model. Based on a hierarchical multi-objective optimisation approach, PhyCLIP will first optimise the primary objective (i.e. cluster as many sequences as possible) before doing so for the secondary objective, that is to cluster as many prior sequences under the same prior cluster as possible, without degrading the solution quality of the primary objective.

To implement the prior, additionally prepare the following tab-delimited prior input file:

```
PRIOR-CLUSTER-ID {tab} Sequence name (must be the same as in tree)
```

```
e.g.
1{tab}seq_A
1{tab}seq_B
2{tab}seq_C
2{tab}seq_D
2{tab}seq_E
...
```

Each prior cluster should come with a unique prior cluster ID (It doesn't necessarily need to be integers so long as they are unique).

If you are using the clustering results from a previous PhyCLIP run, you do **NOT** need to prepare the aforementioned file but just use the output cluster file from that run

(cluster_{gam_method}_{hypo_test}_{sol_index}_{S}_{FDR}_{gamma}_{tree_filename}).txt, see [Output files](#)) as the prior input file.

In your Terminal/Command Prompt:

```
$ phyclip.py --input_file </path/to/input_file.txt> --prior </path/to/prior_input_file.txt>
```

By default, equal weights are given to all input prior clusters (e.g. maintaining that the 2 sequences of prior cluster 1 should be clustered together in the output is given the same importance as doing so for prior cluster 2 comprising of 3 sequences).

If you have justifiable, differential confidence between the input prior clusters, you may also provide weights for the individual prior clusters. To do so, separately prepare the following input:

```
PRIOR-CLUSTER-ID {tab} Weight (int or float)
```

e.g.

```
1{tab}0.2
```

```
2{tab}0.6
```

```
...
```

In your Terminal/Command Prompt:

```
$ phyclip.py --input_file </path/to/input_file.txt> --prior </path/to/prior_input_file.txt> --prior_weights </path/to/prior_weights_input_file.txt>
```

Note that if the weights are given as floats, PhyCLIP expects the summation of weights across all prior clusters to be 1. Otherwise, PhyCLIP will calculate the normalized weights.

Important

1. Note that the final output clusters are secondarily optimised to retain as many sequences of the same prior cluster under the same clade but **NOT** the separateness between these prior clusters. Incorporating the latter may bias the statistical distinction between the clusters, which should be based solely on their differential diversity.
2. If you do choose to run PhyCLIP with a prior, we highly encourage that you perform a separate analysis **without** the prior, compare the results obtained and decide which clustering configuration best answers your research question.