PhyloNet 3 General Overview

1. Introduction

PhyloNet is a tool designed mainly for analyzing, reconstructing, and evaluating reticulate (or non-treelike) evolutionary relationships, generally known as phylogenetic networks. Various methods that we have developed make use of techniques and tools from the domain of phylogenetic trees, and hence the PhyloNet package includes several tools for phylogenetic tree analysis. PhyloNet is released under the GNU General Public License. For the full license, see the file GPL.txt included with this distribution.

1.1 Contributors

PhyloNet is designed, implemented, and maintained by Rice's BioInformatics Group, which is lead by Professor Luay Nakhleh (nakhleh@cs.rice.edu). For more details related to this group please visit http://bioinfo.cs.rice.edu.

2. Installation

System Requirements

In order to run the PhyloNet toolkit, you must have Java 1.7.0 or later installed on your system. All references to the java command assume that Java 1.7 is being used.

- To check your Java version, type "java -version" on your command line.
- To download Java 1.7, please go to website http://www.java.com/en/download/.
- To link to the new downloaded Java 1.7, for mac, try these two commands from command line:
  sudo rm /usr/bin/java
  sudo ln -s /Library/Internet\ Plug-Ins/JavaAppletPlugin.plugin/Contents/Home/bin/java /usr/bin

Downloading phylonet.jar

Acquire the current release of PhyloNet by downloading the most recent version of the PhyloNet JAR file. You will have a file named `PhyloNet_X.Y.Z.jar`, where X is the major version number and Y and Z are the minor version numbers.

Installing the file

Place the jar file in the desired installation directory. The remainder of this document assumes that it is located in `$PHYLONET PATH/jar`. Installation is now complete. In order to run PhyloNet, you must execute the file `PhyloNet_X.Y.Z.jar`, as described in the next section.

3. Basic Usage

The PhyloNet tool is executed by typing the following command into your console:

```
>java -jar $PHYLONET PATH/jar/PhyloNet_X.Y.Z.jar script.nex
```

Where `script.nex` is the NEXUS file containing the commands to be executed.

4. Basic NEXUS Overview

When PhyloNet is invoked with a specified NEXUS script file the tool will execute all of the commands contained within the file's `PHYLONET` block. For example, consider the following NEXUS file:

```
#NEXUS

BEGIN NETWORKS;

Network net = ((a,(b,(c)x#1)),((x#1,d),e));
Network net1 = ((a,(b,(c)x#1)),((d,x#1),e));
Network net2 = ((((a, (c)x#1), d), (b, x)), e);
Network net3 = ((a,b),(c,(d,(e,(f,g)))));
Network net4 = ((f,b),(c,(d,(a,(e,g)))));

END;


BEGIN PHYLONET;

Charnet net -m tree;
Cmpnets net1 net2 -m cluster;
CountCoal net3 net4;

END;
```

Blocks in a NEXUS file start with the `BEGIN` keyword and terminate with the `END;` keyword. This example file contains two blocks--`NETWORKS` and `PHYLONET`. Contained within the `PHYLONET` block is the list of commands PhyloNet will execute when processing the NEXUS file. Commands in a `PHYLONET` block begin with a command identifier and terminate with a semicolon. In this example script, three commands are listed: `Charnet`, `Cmpnets`, and `CountCoal`. Appearing after the command identifier but before the semicolon are any parameters provided to a given command for its execution. For example in the script file above three parameters are provided to the `Charnet` command: `net`, `-m` and `tree`. Details about specific parameters for a given command can be found on the documentation page for the given command.

The `NETWORKS` block provides an area for defining any phylogenetic networks utilized by any command in the `PHYLONET` block. A network definition in the `NETWORKS` block must be of the form:

```
Network "identifier" = "rich newick string";
```

Where *"identifier"* is a user specified name for the network and *"rich newick string"* is a user specified rich newick string. For more information about rich newick strings see its corresponding reference page.

In addition to the `NETWORKS` block, the `TREES` block may also be used to define rich newick strings. However, rich newick strings defined in a `TREES` block may not contain hybridization nodes. Support for the `TREES` block as an alternate declaration to the `NETWORKS` block is for increased compatibility with NEXUS processing tools besides PhyloNet that historically consume or produce `TREES` blocks within NEXUS files.

For example usage of a `TREES` block consider the following NEXUS file:

```
#NEXUS

BEGIN TREES;

Tree speceiesTree = ((((a,b),c),d),e);
Tree geneTree1 = ((((a,b),c),d),e);
Tree geneTree2 = ((a,b),((c,e),d));
Tree geneTree3 = ((a,c),((b,e),d));

END;


BEGIN PHYLONET;

DeepCoalCount {speceiesTree} {geneTree1, geneTree2, geneTree3};

END;
```

The example script's `TREES` block defines four phylogenetic trees in rich newick form that are referenced by the `DeepCoalCount` command within the `PHYLONET` block.

5. Managing Output

The default behavior of PhyloNet for reporting results is to display each command's output on the console. For example given the following NEXUS file `charnet.nex`:

```
#NEXUS

BEGIN NETWORKS;

Network net = ((a,(b,(c)x#1)),((x#1,d),e));

END;


BEGIN PHYLONET;

Charnet net -m tree;

END;
```

we could execute the script by typing the following command on the console:

```
>java -jar $PHYLONET PATH/jar/PhyloNet_X.Y.Z.jar charnet.nex
```

which in turn would append the following output to the console:

```
>java -jar $PHYLONET PATH/jar/PhyloNet_X.Y.Z.jar charnet.nex

Charnet net -m tree
((a,(b,c)),(e,d));
((a,b),((d,c),e));

>
```

Note how the `Charnet` command identifier and its parameters are first displayed followed by the output of the command. This feature becomes very helpful for reading results when many commands are listed in a single NEXUS file.

Most commands, including `Charnet`, support an optional final parameter that specifies the name of a file where command output should be redirected to instead of the console. For example, we could modify `charnet.nex` to send the command's output to the file `C:\temp\charnet_output.txt` as follows:

```
#NEXUS

BEGIN NETWORKS;

Network net = ((a,(b,(c)x#1)),((x#1,d),e));

END;


BEGIN PHYLONET;

Charnet net -m tree "C:\temp\charnet_output.txt";

END;
```

Upon executing the modified script from the console we would now see a new result without any tree results displayed to the console:

```
>java -jar $PHYLONET PATH/jar/PhyloNet_X.Y.Z.jar charnet.nex

Charnet net -m tree "C:\temp\charnet_output.txt"
Writing output to C:\temp\charnet_output.txt

>
```

Opening the file `C:\temp\charnet_output.txt` would reveal the command output:

```
((a,(b,c)),(e,d));
((a,b),((d,c),e));
```

Each command that supports the optional file output parameter in a given NEXUS file can be given its own unique file parameter value resulting in the generation of distinct output files for each command. Alternatively, one may specify the file output parameter for only some commands in a NEXUS file. In this case those commands without the optional parameter would continue to utilize the default behavior and display their results on the console. Repeating a file output parameter value for many commands in a NEXUS file is not advisable as each command's output will overwrite the previous command's output with the same parameter value. One can however send the entire output of a PhyloNet execution to one file by using most operating systems' redirection operators.

5.1 `Nexus_Out` Command

PhyloNet provides a special command called `Nexus_Out` that instructs PhyloNet to additionally create an output NEXUS file containing a copy of each tree result generated by all commands within the script. For example, executing the following NEXUS script:

```
#NEXUS

BEGIN NETWORKS;

Network net = ((a,(b,(c)x#1)),((x#1,d),e));

END;


BEGIN PHYLONET;

Nexus_Out "C:\temp\nexus_out.nex";
Charnet net -m tree;

END;
```

would cause a file `C:\temp\nexus_out.nex` to be generated with contents:

```
#NEXUS

BEGIN TREES;

2_Charnet_1 = ((a,(b,c)),(e,d));

2_Charnet_2 = ((a,b),((d,c),e));

END;
```

in addition to the usual console output for the `Charnet` command.

All trees recorded in the `TREES` section will be of the form:

```
N_Command_M = ...;
```

Where *N* is the number of the command (that is, the nth command) as it appears in the original NEXUS script, *Command* is the command identifier of the command that generated the tree, and *M* denotes the mth tree generated by the nth command.

See Also

- List of PhyloNet Commands