Documentation for **HAL-HAS 2**

*Mixture Models of Nucleotide Sequence Evolution that Account for Rate **H**eterogeneity **A**cross **L**ineages and Rate **H**eterogeneity **A**cross **S**ites*

July, 2024

Disclaimer

Third Party Components

The following third party components are distributed with the Software. You agree to comply with the license terms for these components as part of accessing the Software. Other third party software may also be identified in separate files distributed with the Software.

R : A Computer Language for Statistical Data Analysis version 3.0.1

(http://cran.r-project.org/src/base/R-3/R-3.0.1.tar.gz)

Copyright (C) 2000-2004 The R Core Team

This software is licensed under GNU GPL

JACOBI_EIGENVALUE.C
(http://people.sc.fsu.edu/~jburkardt/c_src/jacobi_eigenvalue/jacobi_eigenvalue.c)

Copyright (C) 2003-2013 John Burkardt

This software is licensed under GNU LGPL (http://www.gnu.org/licenses/lgpl.html)

sse2neon (https://github.com/DLTcollab/sse2neon)

Copyright (c) 2015-2024 SSE2NEON Contributors

This software is licensed under the MIT License

Credits

This software was developed by the bioinformatics and phylogenomics team in Ecosystem Sciences, The Commonwealth Scientific and Industrial Research Organisation (CSIRO), Canberra, Australia.

Introduction

Molecular phylogenetic studies often assume that the evolutionary process behind the divergence of homologous genes was globally stationary, reversible, and homogeneous (SRH) and that a model of evolution comprising one or several site-specific, time-reversible rate matrices (e.g., the GTR rate matrix) is sufficient to accurately model the evolution of the data over the whole tree. However, an increasing body of data suggests that evolution under globally SRH conditions is the exception, rather than the norm. Therefore, we introduce a family of non-stationary and non-homogeneous mixture models that approximate the rate Heterogeneity Across Lineages (HAL) and the rate Heterogeneity Across Sites (HAS) without the assumption of an underlying predefined statistical distribution. We also develop an algorithm for searching model space and identifying a model that is less likely to over- or under-parameterize the data.

Installation of the software

The software was written in C++, and it has been tested under linux and MacOS platform. You need to have C++ compiler and CMake installed in the machine in order to compile the source codes. The compilation steps are shown as follows:

To compile **HAL2** and **HAS2**

```
$ tar -zxvf Hal-Has2-2.7.tar.gz

$ cd Hal-Has2-2.7

$ mkdir build

$ cd build

$ cmake ..

$ make -j
```

Then two executable files will appear:

- HAL2 : The HAL2 program based on normal BIC formula
- HAS2 : The HAS2 program based on normal BIC formula

To compile **HAL2-P** and **HAS2-P**

```
$ tar -zxvf Hal-Has2-2.7.tar.gz

$ cd Hal-Has2-2.7

$ mkdir build

$ cd build

$ cmake -DFLAGS=C_REG ..

$ make -j
```

Then two executable files will appear:

- HAL2-P : The HAL2-P program based on BIC formula with penalty for convergence
- HAS2-P : The HAS2-P program based on BIC formula with penalty for convergence

HAL2 / HAL2-P

Every unique partition of edges represents a HAL model. Since the total number of HAL models is a Bell number, an exhaustive search of all models is infeasible. Therefore, we use an algorithm that searches a subset of the HAL models to identify the optimal and/or near optimal models. HAL includes two algorithms: a bottom-up algorithm, followed by a top-down algorithm.

The following shows the usage of HAL2 / HAL2-P program:

```
Syntax: ./HAL2   <alignment file> <topology file> <other options>
        ./HAL2-P <alignment file> <topology file> <other options>

  <alignment file>     : Multiple alignment file

  <topology file>      : Topology file in Newick format

  -f <format of file>  : The format of the multiple alignment file

                         1 - FASTA format
                         2 - Sequential PHYLIP format (default)

  -u <# of CPUs>       : Number of CPU threads used (default: 4)

  -o <output prefix>   : Prefix for output files
                         (default: <alignment file> w/o .ext)

  -r <checkpoint file> : Resume from the last execution;
                         <checkpoint file>, which was outputted from the last
                         execution of the program, stores all the immediate
                         results (with file extension: '.chkpt.txt')

  -i <# of iterations> : Number of iterations performed for each-time
                         parameter tuning
                         (default: -1 [no limit, run until converge])

  -y <matrix file>     : Optimize the parameters according to the matrix file
                         (no search algorithm will be performed)

  -b <info criteria>   : Information Criteria to be used (default: 3)
                         1 - AIC; 2 - Adjusted IC; 3 - BIC; 4 - CAIC

  -g <gap handling>    : 1 - ignore all columns with gaps;
                         2 - treat gaps as missing data (default)

  -precise             : More precise optimisation, but needs more time.
                         (default: disable)

  -l <start matrix>    : The rate matrix to start from. It is not a file,
                        but a string representing the rate matrix.
                        For example: -l 1,1,1,1,1,2,1,1,2,2,2,2,2,2

  -q                   : 0 - Not run top-down algorithm after bottom-up algorithm
                         1 - Running top-down algorithm after bottom-up algorithm
                         (default: 1)

  -h                   : This help page
```

Output files:

1. <output prefix>.NEW-BU.chkpt.txt, which stores all the intermediate results for resuming the program when necessary.

2. <output prefix>.NEW-BU.<info criteria>.result.txt, which is the resulting optimal rate matrices after the bottom-up algorithm.

3. <output prefix>.NEW-BU.HAL.<info criteria>.result.txt, which is the resulting optimal rate matrices after both the bottom-up and the top-down algorithm.

Example of running HAL2 / HAL2-P program

Example alignment file: data.phy

Example topology file: tree.txt

To execute HAL2 program by using default setting:

```
$ ./HAL2 data.phy tree.txt
```

To execute HAL2-P program by using default setting:

```
$ ./HAL2-P data.phy tree.txt
```

The result file: *data.NEW-BU.HAL.BIC.result.txt*

Difference between HAL2 and HAL2-P

Unlike the original version of HAL program, HAL2/HAL2-P considers the occurrence of convergence (i.e. existence of the same rate matrix along different lineage paths) when computing the optimal arrangement of the rate matrices along the tree.

By default, the information criteria used in HAL2 is BIC. The formula is as follows:

$$BIC = -2\ln(L) + k\,ln(n)$$

where *L* is log-likelihood value, *k* is degree of freedoms, and *n* is the number of sites in the alignment.

However, we found that sometimes the convergences happened more than it should be. Therefore, in HAL2-P, it gives an additional penalty for each occurrence of convergence. The formula of the information criteria being used is:

$$BIC_p = -2\ln(L) + (k + mc)\,ln(n)$$

where *L* is log-likelihood value, *k* is degree of freedoms, *n* is the number of sites in the alignment, *m* is the number of occurrences of convergence, and *c* is a constant to control the magnitude of the penalty. According to the experiments, we found *c*=1 is appropriate. Thus, we set *c*=1 in the program.

HAS2 / HAS2-P

Once the optimal HAL model has been identified, HAS2 / HAS2-P program can be used to group the variable sites into multiple categories (K > 1) and apply the same HAL model to all the rate categories but with different sets of parameters. The value of K is increased, by default, from 1 to 4. HAS2 (or HAS2-P) program uses the BIC (or BIC_p) value to identify the optimal combination of K and HAS model that best fits the data. This model is referred as the optimal HAL-HAS model as it takes into account rate-heterogeneity across lineages and across sites.

The following shows the usage of HAS2 / HAS2-P program:

```
Syntax: ./HAS2   <alignment file> <topology file> <other options>
        ./HAS2-P <alignment file> <topology file> <other options>

  <alignment file>     : Multiple alignment file

  <topology file>      : Topology file in Newick format
```

```
other options:

  -t <HAL result file> : The resulting file from HAL program
                         If no HAL result file is provided, then all edges
                         are assumed to be the same rate group

  -f <format of file>  : The format of the multiple alignment file
                         1 - FASTA format
                         2 - Sequential PHYLIP format (default)

  -u <# of CPUs>       : Number of CPU threads used (default: 4)

  -o <output prefix>   : Prefix for output files
                         (default: <alignment file> w/o .ext)

  -r <checkpoint file> : Resume from the last execution;
                         <checkpoint file>, which was outputted from the last
                         execution of the program, stores all the immediate
                         results (with file extension: '.chkpt.txt')

  -i <# of iterations> : Number of iterations performed for each-time
                         parameter tuning
                         (default: -1 [no limit, run until converge])

  -m <min category #>  : Minimum number of site categories allowed
                         (default: 1)

  -n <max category #>  : Maximum number of site categories allowed
                         (default: 4)

  -w <step category #> : Step of change of site categories
                         (default: 1)

  -precise             : More precise optimzation, but needs more time.
                         (default: disable)

  -b <info criteria>   : Information Criteria to be used (default: 3)
                         1 - AIC; 2 - Adjusted IC; 3 - BIC; 4 - CAIC

  -h                   : This help page
```

There will be two output files:

1. <output prefix>.HAS.chkpt.txt, which stores all the intermediate results for resuming the program when necessary.

2. <output prefix>.HAS.<info criteria>.result.txt, which includes the optimal number of site categories, the corresponding proportion and the details of parameters for each site category.

Example of running HAS2 / HAS2-P program

Example alignment file: data.phy

Example topology file: tree.txt

HAL result file: data.NEW-BU.HAL.BIC.result.txt

To execute HAS2 program:

```
$ ./HAS2 data.phy tree.txt –t data.NEW-BU.HAL.BIC.result.txt
```

To execute HAS2-P program:

```
$ ./HAS2-P data.phy tree.txt –t data.NEW-BU.HAL.BIC.result.txt
```

The result file: *data.HAS.BIC.result.txt*

Contact person

Dr Lars Jermiin

Email: Lars.Jermiin@universityofgalway.ie

Dr Thomas Wong

Email: Thomas.Wong@anu.edu.au