HHSD: Hierarchical heuristic species delimitation

Version 1.0.0, September 2024

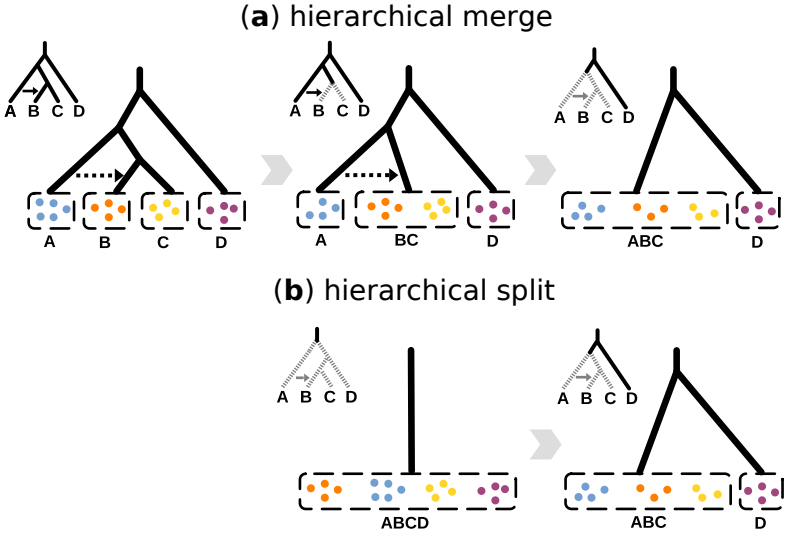
Introduction

HHSD is a pipeline for hierarchical heuristic species delimitation using genomic sequence data under the multispecies coalescent model. The program implements a `merge` algorithm to provide an upper, and a `split` algorithm to provide a lower bound on the number of species.

Background: The iterative algorithms

During each iteration of the `merge` algorithm, the gdi score of all leaf node pairs that share a common ancestor is calculated using parameter estimates from BPP. If the gdi of a given node pair is low (e.g. < 0.2), they are merged into a single species, changing the delimitation. This process of gdi calculation and merging is repeated until none of the leaf nodes pairs have low gdi values. This establishes an empirical maximum bound for the number of species.

To establish an empirical minimum bound, the iterative refinement procedure is conducted in the opposite direction. Given the starting delimitation and guide tree, all of the putative species are first assigned to a single population, and this population is `split` into its constituent populations according to the branching patterns in the guide tree if their gdi is high (e.g > 0.7). The method terminates if no further leaf nodes are found with high gdi values.



Contents

1 Installation

1.1 Prerequisites: correct Python version

Users of HHSD must have Python installed. Supported versions are `3.9`, `3.10`, `3.11` and `3.12` If unsure about the current version of Python, check by typing

```
python --version
```

into the command line and pressing enter. This should result in a similar output to:

```
Python 3.9.16
```

If Python is not installed (typical for Windows), or an older version is installed (this may happen with macOS and Linux releases), check the Python installation documentation.

1.2 Instructions for Linux, macOS, and Windows

In all cases, HHSD is installed on the command line using pip.

Linux

```
git clone https://github.com/abacus-gene/hhsd
```

```
cd hhsd
```

```
pip install .
```

macOS

- Check that you are using the pip version specific to python3, then run:

```
git clone https://github.com/abacus-gene/hhsd
```

```
cd hhsd
```

```
pip3 install .
```

Windows

- If git is aldready installed, run the command:
  ```
  git clone https://github.com/abacus-gene/hhsd
  ```
- If not, download the code by visiting this url. Rename the zip file to `hhsd.zip`, and then uncompress to extract the `hhsd` subdirectory.

Then, install the program as expected:

```
cd hhsd
```

```
pip install .
```

1.3 Checking your install

To check if hhsd was installed successfully, open your command line of choice, and type:

```
hhsd
```

This should result in a greeting message similar to the following:

```
hhsd version 1.1.0
12 cores available
specify control file for analysis with --cfile
```

2 Example analysis using simulated data

HHSD includes a small demonstration dataset that can be used to explore the basic features of the program in a matter of minutes.

How the data was simulated

The dataset simulated here corresponds to the topology presented in the figure on the first page of this manual. Populations **A**, **B**, **C** represent geographical populations of the true species **ABC** with a wide spatial distribution, while **D** is the population representing a new species that split off from the **ABC** species. There is gene flow from **A** to **B**.

Input files for a HHSD analysis

Use the command line to navigate to the relevant folder within the HHSD package.

```
cd hhsd/examples/simulated_abcd
```

The folder contains two input data files, and two control files:

- `MySeq.txt` is a PHYLIP formatted multiple sequence alignment file (MSA) consisting of 100 simulated loci, with two diploid sequences sampled per species per locus (10 sequences/locus), and 500 sites in the sequence.
- `MyImap.txt` is an Imap file used to map the individuals associated with each sequence in the alignment to their five respective populations in the guide tree.
- `cf_sim_merge.txt` is a text based control file that specifies an iterative merge procedure.
- `cf_sim_split.txt` is a text based control file that specifies an iterative split procedure.

2.1 Merge analysis

The merge control file

The text-based control file is used to specify the merge procedure.

```
# output
output_directory = res_sim_merge

# input files
Imapfile = MyImap.txt
seqfile  = MySeq.txt

# guide tree
guide_tree = ((A, (B, C)), D);

# hierarchical algorithm settings
mode = merge
gdi_threshold = <= 0.2, <= 1.0

# migration patterns
migration = {A -> B}
wprior = 10 1

# BPP MCMC settings
threads = 4
burnin =  2000
nsample = 5000
```

- `output_directory` specifies the folder where the results of each iteration (such as the tables of parameter estimates) will be outputted.
- `Imapfile` specifies the location of the Imap.
- `seqfile` specifies the location of the MSA.
- `guide_tree` specifies the phylogeny of the populations in the input dataset.
- `mode` specifies the direction of the iterative process, which in this case will be merging populations in the specified guide tree.
- `gdi_threshold` specifies the gdi values below which a merge proposal will be accepted.
- `threads` specifies the number of cpu threads used during the analysis.
- `burnin` specifies the number of MCMC samples to be discarded as part of the burn-in period.
- `nsample` specifies the number of MCMC samples to be used in the estimation of numeric parameters (tau, theta, W).

Running the merge analysis

To run the analysis specified using the control file, use the following command:

```
hhsd --cfile cf_sim_merge.txt
```

Command line output

The command line feedback should indicate that HHSD performed the following steps:

- Declare starting state: Four Species, **A**, **B**, **C**, and **D**
- Iteration 1: Merge the **A** and **B** populations into **AB**. Three species remain
- Iteration 2: Merge the **AB** and **C** populations into **ABC**. Two species remain.
- Iteration 3: Reject the merge of **ABC** and **D**, and terminate with two species.

For user convenience, during each iteration of the analysis, HHSD outputs the following details to the command line:

- The estimated divergence time (tau) and effective population size (theta) for each population (with 95% HPD intervals).
- *If migration events are present*, the estimated migration rate (M) for each migration event (with 95% HPD intervals).
- The names of node pairs proposed to be merged, their estimated gdi values (again with 95% HPD intervals), and whether the merge proposal was accepted
- The names and phylogeny of the species in the new, modified delimitation

Examining the output files

During each iteration, HHSD creates a corresponding subdirectory in the folder specified by the `output_directory` parameter of the control file. Open `res_sim_merge/Iteration_1`, and explore the contents of the following files to see how they relate to the analysis.

- `estimated_tau_theta.csv`
- `estimated_M.csv`
- `decision.csv`
- `result_imap.txt`
- `result_tree.txt`

2.2 Split analysis

The split control file

The split analysis has a slightly different control file:

```
# output
output_directory = res_sim_split

# input files
Imapfile = MyImap.txt
seqfile  = MySeq.txt

# guide tree
guide_tree = ((A, (B, C)), D);

# hierarchical algorithm settings
mode = split
gdi_threshold = >= 0.7, >= 0.5

# migration patterns
migration = {A -> B}
wprior = 10 1

# BPP MCMC settings
threads = 4
burnin =  2000
nsample = 5000
```

The `output_directory` is updated to be a different folder, `mode` is set to split, and `gdi_threshold` values are adjusted for splitting. For now, do not worry about the specific values of `gdi_threshold`, as a more detailed description of the will be presented in section 4.5.2.

Running the split analysis

To run the analysis specified using the control file, use the following command:

```
hhsd --cfile cf_sim_split.txt
```

Command line output

The command line feedback should indicate that HHSD performed the following steps:

- Declare starting state: One Species, **ABCD**
- Iteration 1: Propose and accept the split of **ABCD** into **ABC** and **D**. We now have two species
- Iteration 2: Propose, but reject the split of the **AB** and **C** populations into **AB**. Terminate the program, with two species remaining.

Examining the output files

Open `res_sim_split/Iteration_2`, and explore the contents of the following files:

- `estimated_tau_theta.csv`
- `decision.csv`
- `result_imap.txt`
- `result_tree.txt`

3 Best practices

3.1 Choosing the guide tree

The guide tree used for HHSD analyses has a fundamental impact on the results. If a widely accepted guide tree is not available for the groups being investigated, the user should use BPP A01 or BPP A11 to infer a guide tree.

3.2 Supplying sufficient data for the MSC+M model

Due to the increase in the number of parameters, reliable inference of migration rate (M) parameters in the MSC+M model requires substantially larger amounts of data, relative to a basic MSC model without migration.

Accordingly, the number of specified migration events should be kept to a minimum, and practitioners should aim to provide as many sequences as they can for populations involved in migration events. To validate that a given dataset is able to constrain the MSC+M model, users should conduct multiple replicate analyses with different migration rate priors and check for the convergence of M parameter estimates

3.3 Interpreting results

Given K input populations, the possible number of species ranges from 1 to K. The merge algorithm establishes an empirical maximum bound (Y) for the number of species, while the minimum bound (X) is established via the split algorithm. As the possible relationships between these quantities is $1 <= X <= Y <= K$, HHSD will often be unable to give an exact estimate for the number of species. In all cases, results from HHSD should be interpreted in the context of additional genetic, morphological, biogeographical, and behavioural results.

4 Detailed documentation of parameters

- ***All parameters marked with '*' must be included in the control file.***
- Any rows of the control file starting with '#', or the parts of rows following '#' will be ignored by the program. This can be used to add comments, or temporarily remove parameter values.
- All file paths in the control file can be relative to the directory where the control file is located, or absolute. In all cases HHSD will always attempt to resolve the absolute path before beginning the analysis.

4.1 File output

4.1.1 output_directory*

`output_directory` specifies the folder where the results of each iteration (such as Imap files, and tables of parameter estimates) will be outputted.

```
output_directory = results_merge
```

4.2 Multiple Sequence Alignment

4.2.1 seqfile*

`seqfile` specifies the location of the PHYLIP formatted multiple sequence alignment (MSA).

```
seqfile = input_multiple_seq_alignment.txt
```

The naming of the sequences within the MSA must follow a specific format, namely:
`sequence_id^individual_id` (e.g. `seq_1^ant320`) or `^individual_id` (e.g. `^ant320`).

```
20 500
seq1^individual_1        TAGAGTCTCC GAGCTCCCAC ATATGTTGTT CTACAATTTC CAAC...
seq2^individual_1        TAGAGTCTCC GAGCTCCCAC ATATGTTGTT CTACAATTTC CAGC...
seq3^individual_2        TAGAGTCTCC GAGCTCCCAC ATATGTGGTT CTACAATTTC CAGC...
seq4^individual_2        TAGAGTCTCC GAGCTCCCAC ATATGTTGTT CTACAATTTC CAAG...
...
```

The MSA used as input to the program can be formatted in a multitude of ways. To account for this, the following parameters can be optionally specified:

4.2.2 cleandata

`cleandata` is specified using a 0-1 flag, e.g.

```
cleandata = 0
```

A value of 1 causes the program to remove all columns in the alignment which have gaps or ambiguity characters, While 0 (the default when the parameter is not specified) means that those will be used in the likelihood calculation.

4.2.3 phase

The `phase` variable is specified using a 0-1 flag, e.g.

```
phase = 0
```

with 0 (the default value if the parameter is not provided) indicating that sequences are fully phased haplotype sequences, while 1 means unphased diploid sequences.

4.3 Specifying the starting delimitation

The starting delimitation is specified in two parts:

4.3.1 Imapfile*

The `Imapfile` parameter specifies the location of the Imap file used to map individuals to their respective populations in the guide tree.

```
Imapfile = Imap_starting.txt
```

On each line of the text-based Imap file, write the individual id and the population id for each respective individual, separated by either a space or a tab:

```
inidvidual_1 population_A
inidvidual_2 population_A
inidvidual_3 population_G
...
```

guide_tree*

The `guide_tree` used for the analysis is given in the parenthesis (Newick) format:

```
guide_tree = (((M,(P,A)),(N,(G,T))),E);
```

The program implicitly names ancestor nodes by concatenating the names of the descendants (the ancestor of nodes A and B will automatically be named AB).

4.4 Specifying the pattern of migration

Specification of migration patterns occurs in two parts:

4.4.1 migration

Migration patterns are specified as a set of migration elements. These individual elements specify the identity of the populations engaging in migration, and the direction of the migration event: `A -> B` for migration from A to B, `A <- B` for migration in the opposite direction, and `A <-> B` for bidirectional migration events. Each element is then separated by ',' and the set of elements is enclosed in '{}' to yield:

```
migration = {A <-> B, B -> C}
```

which is equivalent to:

```
migration = {A -> B, B -> A, C <- B}
```

newline characters following ',' can also add additional separation:

```
migration = {
        A <-> B,
        C <-> B,
        F -> E,
        F -> G
}
```

4.4.2 wprior

`wprior` specifies the gamma prior $\mathbf{G}(\alpha, \beta)$ for all mutation scaled migration rates. This parameter quantifies the average number of changes per site per generation caused by migration.

```
wprior = 10 1
```

In the example above, the migration rate prior is set to **G**(10, 1), with prior mean 10/1 = 10.

Note: Earlier versions of `hhsd` used a different parametrisation.

4.5 Hierarchical algorithm settings

4.5.1 mode*

`mode` specifies the direction of the iterative algorithm. The value can be 'merge' or 'split'

```
mode = merge
```

4.5.2 gdi_threshold*

`gdi_threhold` specifies the gdi values above or below which a split or merge proposal is accepted. A population pair will generate two gdi values (one for each population). Therefore, the user must provide two threshold values as well, separated by a comma.

If `mode` is 'merge', the threshold values must be given in terms of a less-than (denoted <) or less-than-or-equal-to (denoted <=) relation. For example:

```
gdi_threshold = <= 0.2, <= 1.0
```

In this case, a merge proposal will be accepted if at least one of the populations has a gdi value <= 0.2 (while the other population can have any gdi, as all possible gdi values will be less than or equal to 1).

If `mode` is 'split', the thresholds must be given in terms of a greater-than (denoted >) or greater-than-or-equal-to (denoted >=) relation. For example:

```
gdi_threshold = >= 0.7, >= 0.5
```

In this case, a split proposal will be accepted if at least one of the populations has a gdi value > 0.7, while the other population must have a gdi > 0.5.

4.6 MCMC parameters

In total, the MCMC loop consists of $N = burnin + (sampfreq \cdot nsample)$ iterations.

4.6.1 burnin*

`burnin` defines the number of MCMC iterations that are discarded before results are recorded, and can take positive integer value > 200. Typical 'production grade' analyses will set this parameter on the order of 10^4 to 10^5.

```
burnin = 50000
```

4.6.2 nsample*

`nsample` can be specified as any positive integer greater than 1000. Typical analyses will use on the order of 10^5 to 10^6 samples. The computational time for any given analysis scales linearly with `nsample`.

```
nsample = 200000
```

4.6.3 sampfreq

If left empty or unspecified, `sampfreq` defaults to 1. Otherwise, it can be specified as any positive integer. The computational time for any given analysis scales linearly with `sampfreq`.

```
sampfreq = 2
```

4.7 Computational parameters

4.7.1 threads*

In the simplest case, the variable `threads` is used to specify the number of hardware threads used in the underlying BPP process. In general, users should aim to use the maximum amount of threads available on their computer, as this will bring significant speed benefits.

```
threads = 8
```

More complicated assignments to hardware threads can be specified via the sytax:

```
threads = 8 19 1
```

which means BPP will use 8 threads, starting from core/thread 19, with increment 1, so hardware threads 19-36 will be used.

4.7.2 seed

`seed` is the random number seed. If left blank, or unspecified the pipeline will randomly generate a seed and deposit it in the control file, which means that new runs from the same control file can have slightly different results. If you use a positive integer, the program will produce identical results in different runs.

```
seed = 1234
```

4.8 Optional priors

When left unspecified in the control file, the tau and theta priors will be automatically inferred from the sequence data using the method from MINIMALIST BPP.

If the user is concerned about this practice, exact priors can be specified via:

4.8.1 tauprior

`tauprior` specifies the gamma $\mathbf{G}(\alpha, \beta)$ or inverse-gamma prior $\mathbf{IG}(\alpha, \beta)$ for the divergence time parameter for the root in the species tree. Other divergence times are generated from the uniform Dirichlet distribution.

```
tauprior = gamma  3 300
tauprior = invgamma 3 0.03
```

4.8.2 thetaprior

`thetaprior` specifies the gamma $\mathbf{G}(\alpha, \beta)$ or inverse-gamma prior $\mathbf{IG}(\alpha, \beta)$ for the effective population size parameter of all nodes in the species tree.

```
thetaprior = gamma  3 300
thetaprior = invgamma 3 0.05
```

5 Advanced features

5.1 Control file parameter override from the command line

Consider the case when multiple replicate runs are to be conducted under different seeds, or with different priors to ensure convergence. In such cases, all other parameters (except the output directory), should stay consistent among runs. To ensure that such analyses do not require the creation of unnecessary control files, the program features the capability to override control file parameters from the command line.

To override parameters, the `--cfpor` (control file parameter override) option is used. This is then followed by 'parameter = value' tuples separated by commas and a space (", ", commas without a space cannot be used to indicate boundaries between parameters, and will cause errors). An example of how to use this capability correctly would be:

```
hhsd --cfile cf_sim_merge.txt --cfpor seed = 123, output_directory = res_2
```

When running the with parameter overrides, the program will inform the user:

```
< Checking control file arguments... >

The following control file parameters have been overridden via --cfpor:
seed = 123
output_directory = res_2'
```

This feature also enables replicate or alternate analyses to be automatically specified using a shell (bash, zsh, powershell) script!

5.2 gdi estimation mode

Users wanting to get gdi estimates from the program, rather than delimitation results can set the `gdi_treshold` value to 'None':

```
gdi_threshold = None
```

When launching an analysis this way, the program will notify the user:

```
...
Activating gdi estimation mode. All proposals will be automatically accepted!
...
```

The estimates for the gdi values of all populations (not just the leaf nodes) in the guide tree can then be found by examining the `decision.csv` files for each iteration.