HHSD: Hierarchical heuristic species delimitation

Version 0.9.8, September 2023

Introduction

HHSD is a python pipeline for hierarchical heuristic species delimitation using genomic sequence data under the multispecies coalescent model. Given K input populations, the possible number of species ranges from 1 to K. The program implements a `merge` algorithm to provide an upper, and a `split` algorithm to provide a lower bound on the number of species.

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 3.9+ installed. Check your current version with:

```
python --version
```

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 instructions for your operating system here.

1.2 Instructions for Linux, macOS, and Windows

1.2.1 Linux

Installation on Linux can be accomplshed via the following set of commands:

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

```
cd hhsd
```

```
pip install .
```

1.2.2 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 .
```

1.2.3 Windows

Windows installations are slightly more complicated.

- 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 0.9.8
12 cores available
specify control file for analysis with --cfile
```

2 Example analyses

2.1 A quick introduction using simulated data

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

2.1.1 Why and how the data was simulated

This simulated dataset was originally introduced by Leache et al (2019), to demonstrate how a hierarchical delimitation approach based on the gdi, could help reduce oversplitting in species delimitation analyses. There are two 'true' species in the data, split into five populations. Populations **A**, **B**, **C**, **D** represent geographical populations of the first species with a wide spatial distribution, while **X** is the population representing a new species that split off from population **A**. There is extensive gene flow between any two neighbouring populations of species **ABCD**, whereas there is no gene flow to or from **X**.

2.1.2 Input Data Files

To access the relevant files, start by using the command line to navigate to the relevant folder within the HHSD package.

```
cd hhsd/examples/simulated_abcdx
```

The folder contains three files, two of which will be utilised by the program as input data, and one is used to set up the procedure.

- `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.

2.1.3 The control file

```
# output
output_directory = res_sim_merge

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

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

# hierarchical algorithm settings
mode = merge
gdi_threshold = <0.2

# BPP MCMC settings
threads = 4
burnin =  500
nsample = 2000
```

- `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 value 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.
- `nsample` specifies the number of MCMC samples to be used in parameter estimation.

2.1.4 Running the analysis and examining the command line output

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

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

During each iteration of the analysis, HHSD outputs the following to the command line:

- The estimated divergence time (tau) and effective population size (theta) for each population (with 95% confidence intervals).
- The names of node pairs proposed to be merged, their estimated gdi values, and whether the merge proposal was accepted
- The names and phylogeny of the species in the new, modified delimitation

2.1.5 Examining the output files

During each iteration, HHSD creates a corresponding subdirectory in folder specified by the `output_directory` parameter of the control file (e.g `res_sim_merge/Iteration_1`). Each such subfolder contains:

- `estimated_tau_theta.csv` The tau and theta values for each population (with 95% confidence intervals).
- `decision.csv` The names of node pairs proposed to be merged, their estimated gdi values, and whether the merge proposal was accepted.
- `result_imap.txt` The assignment of individuals to the new species generated by the iteration.
- `result_tree.txt` The phylogeny of the new species generated by the iteration.

As expected, HHSD is able to recover the correct two species delimitation.

2.2 Empirical data from the Milksnake Lampropeltis triangulum

This second demonstration seeks to showcase the ability of HHSD to perform delimitation analysis in systems involving migration. This demo will take multiple hours to run. As most details are identical to the simpler case above, only the differences resulting from the specification of migration events will be discussed.

2.2.1 Biological Background

The American milksnake *Lampropeltis triangulum* is a New World snake with one of the widest known geographic distributions within the squamates. There are seven populations (subspecies) engaging (*abnorma*, *polyzona*, *micropholis*, *triangulum*, *gentilis*, *annulata*, and *elapsoides*), and the species status of each population is highly contested due to evidence of genetic exchange between neighbouring subspecies (Chambers & Hillis 2020).

2.2.2 Data Files

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

```
cd hhsd/examples/empirical_milksnake
```

The folder contains four files, two of which will be utilised by the program as input data, and two are alternative control files corresponding to the merge and split analyses respectively.

- `MSA_Lampropeltis.txt` MSA
- `Imap_Lampropeltis.txt` Imap file
- `cf_milksnake_merge.txt` specifies an iterative merge procedure
- `cf_milksnake_split.txt` specifies an iterative split procedure

2.2.3 Control file

```
output_directory = res_milksnake_merge

Imapfile = Imap_Lampropeltis.txt
seqfile = MSA_Lampropeltis.txt

guide_tree = (((Mi, (Po, Ab)), (An, (Ge, Tr))), El);

# migration events and priors
migration = {
        Po <-> Ab,
        Po <-> An,
        An <-> Ge,
        Ge <-> Tr,
        Ge <-> El,
        Tr <-> El,
}
migprior = 0.1 10

mode = merge
gdi_threshold = <0.3

threads = 8
burnin = 50000
nsample = 200000
```

The only new parameters in this analysis relate to the specification of migration events:

- `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` represent bidirectional migration events between two populations.
- `migprior` specifies the default gamma prior for all migration rates. This value is set to $\mathbf{G}(0.1, 10)$, with prior mean $0.1/10 = 0.01$ immigrants per generation.

In the second `cf_milksnake_split.txt` file, the only difference is the value of the `mode` and `gdi_threshold` parameters, which are set to their split-specific values.

```
output_directory = res_milksnake_merge

Imapfile = Imap_Lampropeltis.txt
seqfile = MSA_Lampropeltis.txt

guide_tree = (((Mi, (Po, Ab)), (An, (Ge, Tr))), El);

# migration events and priors
migration = {
        Po <-> Ab,
        Po <-> An,
        An <-> Ge,
        Ge <-> Tr,
        Ge <-> El,
        Tr <-> El,
}
migprior = 0.1 10

mode = split
gdi_threshold = >0.7

threads = 8
burnin = 50000
nsample = 200000
```

2.2.4 Running the analyses and examining the output

The analyses should be run in sequence, letting the first finish before starting the next.

```
hhsd --cfile cf_milksnake_merge.txt
```

```
hhsd --cfile cf_milksnake_split.txt
```

The command line output of HHSD will contain an additional section (on top of the three discussed in the simulated demo). This details the estimated migration rate (M) for node pairs participating in migration events.

Similarly, checking the iteration subdirectories (e.g `res_milksnake_merge/Iteration_1`) will reveal the presence of an additional file, `estimated_M.csv`, containing the estimated migration rates (with 95% confidence intervals) for the relevant population pairs.

The analyses should suggest that the upper bound for the number of species is three, while the lower bound is a single species. General guidelines for interpreting results are discussed in the following section.

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

When analysing populations engaging in migration events, the user has the option of taking these into account in the delimitation analyses. However, 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 with only tau (τ) and theta (θ) parameters.

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 the current dataset is able to constrain the MSC+M model, users should conduct multiple replicate analyses with different migration rate priors. In situations where the phylogenetic information is insufficient, the inferred migration rates in the analyses will simply be the prior mean values.

3.3 Interpreting results

As discussed in the introduction, 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

General advice:

- ***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 means that those will be used in the likelihood calculation. This is the default behaviour when `cleandata` is unspecified

4.2.3 phase

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

```
phase = 0
```

with 0 (the default) indicating that sequences are fully phased haplotype sequences, while 1 means unphased diploid sequences. If this line is missing, all sequences are assumed to be fully phased (that is, flag 0 for all species).

4.3 Specifying the starting delimitation

The starting delimitation is specified in two parts, using a guide tree to specify the phylogeny of the starting populations, and an Imap file to specify the assignment of individuals in the MSA to populations.

4.3.1 Imapfile*

`Imapfile` 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 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
inidvidual_4 population_N
...
```

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);
```

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 of course 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 migprior

`migprior` specifies the gamma prior $\mathbf{G}(\alpha, \beta)$ for all migration rates. Migration is measured in the number of migrant individuals per generation.

```
migprior = 2 20
```

In the example above, the migration rate prior is set to $\mathbf{G}(2, 20)$, with prior mean $2/20 = 0.1$.

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 value above or below which a split or merge proposal is accepted. If `mode` is 'merge', the threshold value must be given in terms of a less-than relation.

```
gdi_threshold = <0.3
```

Similarly, if `mode` is 'split', the threshold value must be given in terms of a greater-than relation

```
gdi_threshold = >0.7
```

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. Having sufficient burnin time is crucial. `burnin` can be specified as any positive integer greater than 200. Typical analyses will set this parameter on the order of 10^4 to 10^5. The computational time for any given analysis scales linearly with `nsample`.

```
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:

```
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.

In general, users should aim to use the maximum amount of threads available on their computer, as this will bring significant speed benefits

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

As in all Bayesian analyses, the BPP A00 procedure used for the estimate of numeric parameters requires that a prior distribution is specified. While migration rate priors must be supplied by the user, tau (τ) and theta (θ) priors, when not specified in the control file will be automatically inferred from the sequence data. The procedure is identical to the established program MINIMALIST BPP.

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

4.8.1 tauprior

`tauprior` specifies the inverse-gamma prior $\mathbf{IG}(\alpha, \beta)$ for the divergence time parameter for the root in the species tree. The mean of this distribution is $\beta/(\alpha - 1)$. Other divergence times are generated from the uniform Dirichlet distribution.

```
tauprior = 3 0.03
```

4.8.2 thetaprior

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

```
thetaprior = 3 0.056
```

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. For example:

```
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 not satisfied with the results from the merge or split algorithms, or those simply 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.