HHSD

Daniel Kornai

2023

Contents

0.1 Introduction

0.2 Installation

0.2.1 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.

0.2.2 Instructions for Linux, macOS, and Windows

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 .
```

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

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 .
```

0.2.3 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 0.9.7
12 cores available
specify control file for analysis with --cfile
```

0.3 Examples

0.3.1 Simulated data from:

0.3.2 Empirical data from the Milksnake Lampropeltis triangulum

Biological Background

Files

Control file

The control file for the merge analysis is:

```
# output
output_directory = res_milksnake_merge

# input files
Imapfile = Imap_Lampropeltis.txt
seqfile = MSA_Lampropeltis.txt

# guide tree
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

# hierarchical algorithm settings
mode = merge
gdi_threshold = <0.3

# BPP MCMC settings
threads = 8
burnin = 200000
nsample = 500000
```

`output_directory` specifies the folder where the results of each iteration (such as Imap files, and tables of parameter estimates) will be outputted. `Imapfile` specifies the location of the Imap file used to map individuals to their respective populations in the guide tree. `seqfile` specifies the location of the PHYLIP formatted multiple sequence alignment (MSA). The `guide_tree` used for the analysis is given in the Newick format. `migration` is used to specify bidirectional migration events between geographically adjacent populations, while `migprior` specifies a gamma prior for migrations rates with mean = 0.01.

Running the analysis

Examining the results

0.4 Detailed explanation of parameter syntax

- ***All parameters marked with '*' must be included in the control file.***
- 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.

0.4.1 File output

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
```

0.4.2 Multiple Sequence Alignment

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:

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

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).

0.4.3 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.

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, ending with a semicolon (;):

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

0.4.4 Specifying the pattern of migration

Specification of migration patterns occurs in two parts:

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
}
```

migprior

`migprior` parameter specifies the default gamma prior for all migration rates:

```
migprior = 2 20
```

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

0.4.5 MCMC parameters

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

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
```

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
```

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
```

0.4.6 Computational parameters

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

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
```

0.5 Best practices

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.

Hidden prior estimation of τ and θ

As in all Bayesian analyses, The BPP A00 procedure used for intra-model inference of numeric parameters requires that a prior distribution is specified. While migration rate priors must be supplied by the user, τ and θ priors, when not specified in the master 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, they should conduct analyses under multiple different priors, and check for convergence.

Ensuring run-to-run reproducibility

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 τ and 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. In situations where the phylogenetic information is insufficient, the inferred migration rates in the analyses will simply be the prior mean values. 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.

0.6 Advanced features

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), will 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.

```
hhsd --cfile giraffe_merge.txt
```

To override certain parameters, the `--mcfpor` (master control file parameter override) option is used, which is followed by parameter name - parameter value pairs, separated by commas:

```
hhsd --cfile giraffe_merge.txt --cfpor seed = 123, migprior = 0.01 10
```

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
migprior = 0.01 10
```

This feature enables replicate analyses to be specified using a bash script. See the included `giraffe.sh` demonstration file for an example.