G-PhoCS – Generalized Phylogenetic Coalescent Sampler
version 1.3

Contents

1. About *G-PhoCS*

G-PhoCS is a software package for inferring ancestral population sizes, population divergence times, and migration rates from individual genome sequences. ***G-PhoCS*** accepts as input a set of multiple sequence alignments from separate neutrally evolving loci along the genome. Parameter inference is done in a Bayesian manner, using a Markov Chain Monte Carlo (MCMC) to jointly sample model parameters and genealogies at the input loci.

G-PhoCS is derived from and inspired by ***MCMCcoal*** developed by Ziheng Yang. The original inference framework introduced in MCMCcoal was extended in several ways to make it appropriate for inference of population history rather than species history.

This user manual provides basic information for users of the software. Users should make sure to carefully read the manual before trying out the software. Sample files are also provided in the package. Additional important details about the method are provided in the supplement of the ***G-PhoCS*** paper: Gronau I, Hubisz MJ, Gulko B, Danko CG, Siepel A. Bayesian inference of ancient human demography from individual genome sequences. *Nature Genetics* **43** 1031–1034. 2011.

The most relevant sections in the supplement are Section 1 (preparing data for analysis), section 4 (the ***G-PhoCS*** algorithm), and Section 5 (practical execution tips).

For questions, comments, and feature requests for ***G-PhoCS***, please contact Ilan Gronau at: ilan.gronau@idc.ac.il

Good luck,
Ilan.

2. Download and install

G-PhoCS is developed on Linux, but can be compiled on any system. For the multi-threaded parallel version you will need to have a C compiler with OpenMP library installed.

1. Clone the ***G-PhoCS*** source code from the Github repository
   ```
   git clone https://github.com/gphocs-dev/G-PhoCS.git
   ```
2. Move to the obtained directory
   ```
   cd G-PhoCS
   ```
3. Compile ***G-PhoCS***. See Section 9 for multithreading support.
   ```
   make
   ```

The ***G-PhoCS*** binaries (G-PhoCS and readTrace) can be now found in the bin/ subdirectory.

4. Post-Install Test Run
   ```
   bin/G-PhoCS sample-control-file.ctl
   ```

3. Latest updates

The main updates in version 1.3 include:

- Introductoin of a multi-threaded implementation, which allows reducing running time on multi-core CPUs. See details here.
- A control file generator Java applet for constructing setup files for ***G-PhoCS*** analysis. See details here.

The main updates in version 1.2.3 include:

- enabling analysis of ancient DNA samples by associating a sample age parameter with each ancient sample. See 'age' attribute in CURRENT-POP.

4. Overview of *G-PhoCS* analysis: input and output

When preparing your data for analysis by ***G-PhoCS***, you will need to create a sequence file and a control file. The sequence file contains your sequence data, and the control file contains the specification for the prior distribution over model parameters and instructions for the sampler.

The main output of the MCMC sampler is produced in the **trace file** (whose path is specified in the control file). The first column in the trace file contains the number of MCMC iteration from which the sample is given (starting from 0). After that, there is a column for every model parameter (effective population sizes, population divergence times, migration rates). The last two columns correspond to the average data log likelihood (across all loci), and the full log likelihood (summed across loci, and considering the genealogy priors). All parameters are given with accuracy of 10^{-5}. In order to obtain optimal relative accuracy, parameters can be printed in any factor (by setting the appropriate attributes in the control file). The trace file can be analyzed using the readTrace binary supplied with ***G-PhoCS***. The trace file is also formatted so that it could be viewed by ***Tracer*** from the ***BEAST*** package.

Sample	theta_A	theta_B	theta_C	theta_D	theta_AB	theta_ABC	theta_root	tau_AB	tau_ABC	tau_root	m_D->B	Data-ld-ln	Full-ld-ln
0	1.06147	1.02213	0.85924	1.09034	0.99918	0.91122	1.11978	0.04961	0.09858	0.54487	0.00000	-1345.658845	-1407136.676
1	1.09477	0.96538	0.85924	1.12269	0.99918	0.93931	1.11978	0.04961	0.09858	0.54487	20.24997	-1343.305172	-1405052.121
2	1.03987	0.96538	0.90605	1.12269	0.99918	0.93931	1.11978	0.04961	0.09797	0.54487	19.83021	-1342.676200	-1404146.180
3	0.99947	1.00241	0.90897	1.16359	1.00241	1.00738	1.12339	0.04977	0.09829	0.57991	19.18764	-1342.502793	-1403567.133
4	1.04238	1.02384	0.95178	1.16597	0.97595	1.03361	1.12569	0.04987	0.09849	0.58110	19.44754	-1342.036533	-1403214.705
5	1.08084	1.07690	0.95178	1.20241	0.94602	1.03361	1.12569	0.05291	0.09849	0.58110	19.27217	-1342.093022	-1403128.813
6	1.12640	1.11818	0.95178	1.24548	0.91416	1.03361	1.12569	0.05291	0.09849	0.58110	19.67596	-1341.907213	-1403025.000
7	1.08826	1.06609	0.95666	1.31251	0.95618	1.03891	1.18004	0.05318	0.09899	0.60630	19.88259	-1342.166514	-1402881.053
8	1.08826	1.03309	0.92669	1.31251	0.95618	1.06409	1.22692	0.05318	0.09899	0.60630	19.50578	-1342.120212	-1402792.119
9	1.03226	1.08487	0.99071	1.37284	0.92455	1.06409	1.26614	0.05318	0.09899	0.60630	19.16283	-1342.138429	-1402730.678
10	1.07730	1.14046	0.95068	1.45034	0.95010	1.06409	1.26614	0.05318	0.09899	0.60630	18.88338	-1342.148108	-1402704.529

An example trace file for ten iterations of the sample run

G-PhoCS also outputs a summary log onto the standard output. The output log starts with the ***G-PhoCS*** banner (with version and date indicated). Then there is possible information about the control file and sequence file used. After that, there is some information about the MCMC (number of iterations, burn-in, etc.). A more detailed version can be obtained by running ***G-PhoCS*** with the -v option. After this, the log contains an update on the progress of the MCMC. The status line is updated every X MCMC iterations (X is specified in the control file by the `iterations-per-log` attribute), and a new log line is generated every Y logs (Y is specified in the control file by the `logs-per-line attribute`). The columns of the log mostly describe acceptance rates of the various updates of the MCMC (see Tuning the MCMC) and are ordered as follows:

- column 1 – the number of MCMC iterations (samples)
- columns 2–4 – the acceptance rate of the local update steps of the MCMC (a local update step is one that modifies a single genealogy).
- Columns 5–6 – the acceptance rate of the updates for model parameters for the effective population sizes and migration rates
- columns 7–X – the acceptance rates of the updates associated with each ancestral population divergence time in the model.

- column X+1 – the rejection rate of divergence time updates attributed to migration-related rubber-band conflicts (RbberBnd). Such a rejection is caused by a migration event that is moved outside its feasible range due to the rubber-band operation performed during a divergence time update (see Section 4 and Figure E of the paper supplement).
- columns X+2–X+3 the acceptance rates for locus-specific mutation rate (if variable rates model is not used, these are 0), and the scaling (mixing) update.
- column X+4 – the average recorded data log likelihood across all loci and across all MCMC iterations during the time of log.
- Column X+5 – time since program started HH:MM:SS

```
**************************************************************
*                 G-Phocs ver 1.3     Jan 2017               *
**************************************************************
Setting Maximum Thread Count to: 1
Reading control settings from file sample-control-file.ctl...
Done.
Reading sequence data...  1000 loci, as specified in sequence file seqs-sample.txt.
Reading loci (.=100 loci): ..........

Starting MCMC: 0 burnin, 5000 running, sampled every 0 iteration(s).
There are 11 parameters in the model.
Samples   CoalTimes MigTimes  SPRs      Thetas    MigRates TAU_ 4    TAU_ 5    TAU_ 6    RbberBnd  MutRates  Mixing    | DATA-ln-ld |  TIME
-------------------------------------------------------------------------------------------------------------------------------------------
    500    64.5%     96.2%     69.1%     63.9%     98.0%     23.5%     23.5%      2.0%       0.0%      0.0%     96.1%    |-1348.092958|  0:31
   1000    64.1%     97.1%     69.9%     67.2%     94.1%      3.9%      3.9%     11.8%       0.7%      0.0%     96.1%    |-1348.113581|  1:01
   1500    64.2%     98.2%     69.8%     68.1%     98.0%      7.8%      0.0%     11.8%       0.0%      0.0%     94.1%    |-1348.206668|  1:33
   2000    65.2%     97.4%     69.5%     67.5%     98.0%      0.0%      0.0%      7.8%       0.0%      0.0%     96.1%    |-1347.278286|  2:06
   2500    64.0%     97.6%     69.7%     67.2%     90.2%      7.8%      3.9%     17.6%       0.0%      0.0%     92.2%    |-1347.845674|  2:35
   3000    64.5%     94.5%     69.4%     63.9%     90.2%     19.6%     11.8%     11.8%       0.0%      0.0%     94.1%    |-1347.915382|  3:04
   3500    64.2%     97.4%     69.6%     70.6%     96.1%     21.6%     25.5%      7.8%       0.0%      0.0%     92.2%    |-1348.376035|  3:33
   4000    64.5%     98.7%     69.8%     68.1%     94.1%     49.0%     47.1%      3.9%       0.0%      0.0%     96.1%    |-1347.770289|  4:03
   4500    64.0%     98.0%     69.7%     65.3%     94.1%      3.9%      3.9%     15.7%       0.0%      0.0%     94.1%    |-1348.385142|  4:33
   5000    64.6%     98.3%     69.4%     68.1%     98.0%      2.0%      5.9%      9.8%       0.0%      0.0%     96.1%    |-1347.663113|  5:02

MCMC finished. Time used:  5:02
```

An example log for the sample run

5. The sequence file

The sequence file is a simple text file indicating the sequence alignments to be analyzed. Blank lines are ignored in this file, and different fields within a line should be separated by any type of whitespace. The first line contains the number of loci in the file. If L loci are indicated at the file header, but more than L loci are specified in the file, only the first L loci will be read. If less than L loci are present, then an error message will be given. After this line, there is a block for each locus. The first line in the block has three fields indicating the locus name (no more than 50 characters), the number of sequences N to read, and the sequence length K. The locus block then contains exactly N additional non-empty lines. Each line has two fields: a sample name for which this sequence is to be associated, and the sequence. The sequence must be exactly of length K, and must contain only IUPAC characters for DNA bases: `A,C,G,T,R,Y,S,W,K,M,` and `N`. Characters `R,Y,S,W,K,M` indicate heterozygous genotypes, and can only be used when the sample is specified as a diploid sample in the control file. Character N indicates missing (or masked) genotype. No other character is allowed (e.g., `U, -, ?`, etc.), and the sequence is case-insensitive.

```
1000

locus1 6 1000
one   CCAGAGAGCaCGTAGTTCCGGTAATGGTTGTCCACATGGTAACGGACTCCGACGGAAGAGACCATCATAAATTAGATGCATAGGGCATGAGCCCATCGAGAAGAGGATTGAACTCCCACCTCTCGGGGGCTTCCT...
two   CCAGAGAGCTCGTAGTTCCGGTAATGGTTGTCCACATGGTAACGGACTCCGACGGAAGAGACCATCATAAATTAGATGCATAGGGCATGAGCCCATCGAGAAGAGGATTGAACTCCCACCTCTCGGGGGCTTCCT...
three CCAGAGAGCTCGTAGTTCCGGTAATGGTTGTCCACATGGTAACGGACTCCGACGGAAGAGACCATCATAAATTAGATGCATAGGGCATGAGCCCATCGAGAAGAGGATTGAACTCCCACCTCTCGGGGGCTTCCT...
four  CCAGAGAGCTCGTAGTTCCGGTAATGGTTGTCCACATGGTAACGGACTCCGACGGAAGAGACCATCATAAATTAGATGCATAGGGCATGAGCCCATCGAGAAGAGGATTGAACTCCCACCTCTCGGGGGCTTCCT...
five  CCAGAGAGCTCGTAGTTCCGGTAATGGTTGTCCACATGGTAACGGACTCCGACGGAAGAGACCATCATAAATTAGATGCATAGGGCATGAGCCCATCGAGAAGAGGATTGAACTCCCACCTCTCGGGGGCTTCCT...
six   CCAGAGAGCTCGTAGTTCCGGTAATGGTTGTCCACATGGTAACGGACTCCGACGGAAGAGACCATCATAAATTAGATGCATAGGGCATGAGCCCATCGAGAAGAGGATTGAACTCCCACCTCTCGGGGGCTTCCT...

locus2 6 1000
...
```

An example header of the sample sequence file

6. The control file

The control file allows the user to specify to ***G-PhoCS*** the model to use in its analysis, as well as other controls that affect the MCMC and output format. A control file typically has four modules (in this order): `GENERAL-INFO`, `CURRENT-POPS`, `ANCESTRAL-POPS`, and `MIG-BANDS`. The Control File Generator (CFG) is a graphical application that you may use to help set up the control file (see Section 11 for more details).

Some useful guidelines in building a control file: Each line in the file either defines the start or end of a module (or sub-module), or sets a certain attribute. In that case, it contains the attribute name and a value (or list of values, in some cases). Some attributes are obligatory, while others are optional (with default settings). The order of the different attributes within a module does not matter. Empty lines and white spaces are ignored, and any part of a line after a '#' character is also ignored (can be used for documentation).

GENERAL-INFO

The `GENERAL-INFO` module contains essentials arguments for running ***G-PhoCS***. Below, we give a detailed list of all attributes one can set in this module. More information on setting up an effective MCMC run is given in the section titled Tuning the MCMC.

```
GENERAL-INFO-START

       seq-file             seqs-sample.txt
       trace-file           mcmc.out
       mcmc-iterations      1000000

       # automatic finetune finder
       find-finetunes       TRUE

       tau-theta-alpha      1.0
       tau-theta-beta       10000.0

       mig-rate-alpha       0.002
       mig-rate-beta        0.00001

 GENERAL-INFO-END
```

An example of a basic `GENERAL-INFO` module

`GENERAL-INFO` attribute list:

`seq-file`	Name of file containing sequence alignments for analysis (see previous section) **Obligatory**
`trace-file`	Name of file onto which trace will be written (see section on trace analysis) **Optional**, default: "`mcmc-trace.out`"
`num-loci`	Number of Loci to read from sequence alignment file. Will read first N loci. **Optional**, default: read all loci in file
`random-seed`	Seed for random number generator used in ***G-PhoCS*** **Optional**, default: use value initialized by current time stamp
`burn-in`	Number of MCMC iterations performed before tracing to output. It is recommended to run the MCMC with a pre-set burn-in of 0, and to apply the actual burn-in only when post-processing the trace file (by ignoring the top part of the file). **Optional**, default: `0`
`mcmc-iterations`	Number of MCMC iterations performed after burn-in **Optional**, default: `10,000`
`mcmc-sample-skip`	Number of iterations between each two traced samples **Optional**, default: `0` (means each sample is traced)
`start-mig`	Iteration at which migration starts being sampled (it is often useful to let the MCMC converge before introducing migration into the model **Optional**, default: `0`
`iterations-per-log`	Number of MCMC iterations between each logging to stdout **Optional**, default: `100`
`logs-per-line`	Number of log updates to write in one line to stdout **Optional**, default: `100`
`tau-theta-print`	A multiplicative factor to use when printing τ and θ parameters to trace file **Optional**, default: `1.0`
`tau-theta-alpha` `and` **`tau-theta-beta`**	α and β parameter for the Gamma distribution used for priors of all τ and θ parameters **Obligatory**, unless the priors of all τ and θ parameter are individually set
`mig-rate-print`	A multiplicative factor to use when printing migration rates to trace file **Optional**, default: `1.0`
`mig-rate-alpha` `and` **`mig-rate-beta`**	α and β parameter for the Gamma distribution used for priors of all migration rates **Obligatory**, unless the priors of all migration rates are individually set
`locus-mut-rate`	Determines the model used for rate variation across loci. `CONST` – constant rate `VAR` – variable rate across loci. The α parameter for Dirichlet prior for rate variation should be specified after `VAR`. E.g., locus-mut-rate `VAR` 1.0 `FIXED` – fixed rates. The name of a file containing relative mutation rates for all loci should be specified after `FIXED`. E.g., locus-mut-rate `FIXED` rate-

	file.txt. The rate file should contain a list of floating point numbers, separated by white spaces, indicating relative rates for the various loci, according to the order of appearance of these loci in the sequence file. **Optional**, default: `CONST`
`finetune-coal-time`	Finetune parameter for the MCMC update of coalescence times **Obligatory**, unless find-finetunes is set to `TRUE`.
`finetune-mig-time`	Finetune parameter for the MCMC update of migration event times **Obligatory**, unless find-finetunes is set to `TRUE`.
`finetune-theta`	Finetune parameter for the MCMC update of all θ parameters **Obligatory**, unless find-finetunes is set to `TRUE`.
`finetune-mig-rate`	Finetune parameter for the MCMC update of all migration rates **Obligatory**, unless find-finetunes is set to `TRUE`.
`finetune-tau`	Finetune parameter for the MCMC update of all τ parameters **Obligatory**, unless find-finetunes is set to `TRUE`, or the finetune parameters of all τ parameters are set individually
`finetune-locus-rate`	Finetune parameter for the MCMC update of locus-specific mutation rates **Obligatory** when locus-mut-rate is `VAR`, unless find-finetunes is set to `TRUE`.
`finetune-mixing`	Finetune parameter for the MCMC scaling update (called 'mixing') **Obligatory**, unless find-finetunes is set to `TRUE`.
`find-finetunes`	When set to `TRUE` the automatic finetune finding procedure is invoked **Optional**, default: `FALSE`
`find-finetunes-num-steps`	The number of update steps for the automatic finetune finding procedure **Optional**, default: `100`
`find-finetunes-samples-per-step`	The number of MCMC iterations per finetune update in the automatic finetune finding procedure **Optional**, default: `100`
`no-mixing`	If this optional flag is set, then mixing update step is not applied. Used with arbitrary value (e.g. **`no-mixing TRUE`**). Note that mixing is also not applied if one of the sampled populations is associated with a fixed non-zero age (see below). (v1.2.3 and up)

CURRENT-POPS

The `CURRENT-POPS` module contains specification of the samples used for analysis, and their assignment to different populations. Other parameters associated with these populations may also be specified. The `CURRENT-POPS` module consists of a list of `POP` sub-modules – one for each of the sampled populations. The central part of the `POP` sub-module is the samples line. A samples line starts with the token '`samples`', and then has two fields for each sample analyzed in that population: a sample name, and a character `h/d` indicating whether it is a haploid or a diploid sequence. Sample name should be identical to the one used in the sequence file (same capitalization too). Sequences associated with a sample indicated as haploid (h) can have only genotypes of type `A,C,G,T,` and `N`. Sequences associated with a sample indicated as diploid (d) will be analyzed using the phasing integration technique implemented in ***G-PhoCS***.

Sequences in the sequence file associated with samples that are not mentioned in the control file will be ignored. A sample indicated in the control file does not have to have a sequence associated with it at every locus, but is expected to have at least one sequence in one locus associated with it (otherwise, an error message is given). Other attributes that can be set in the `POP` sub-module are listed below.

```
CURRENT-POPS-START

        POP-START
                name            A
                samples         one d
        POP-END

        POP-START
                name            B
                samples         two d
        POP-END

        POP-START
                name            C
                samples         three d
        POP-END

CURRENT-POPS-END
```

An example of a basic `CURRENT-POPS` module

`POP` attribute list (for sampled population):

`name`	Name of the sampled population (50 characters max) **Obligatory**
`samples`	Sample list (see description above) **Obligatory**
`theta-alpha` and **`theta-beta`**	α and β parameter for the Gamma distribution used for prior of the θ parameter associated with the population **Optional**, default: global values set for all τ and θ parameters in the `GENERAL-INFO` module
`theta-print`	A multiplicative factor to use when printing θ parameter to trace file **Optional**, default: global value in the `GENERAL-INFO` module
`age`	Specify age for all samples in the populations. To be used for ancient DNA samples. This attribute must be given with a number specifying a starting age for the sampler and a flag ('e' or 'f') specifying whether the sampler should keep the age fixed (f) or estimate it (e). The sample age will appear in the trace in both cases, unless it is set to a fixed value of 0.0. If one of the current pops is given a positive fixed sample age, then the mixing update step is disabled in the sampler (because this step scales all parameters). **Optional**, default: sample age is set to 0 and fixed. (v1.2.3 and up) **Example1:** `age 0.00045 f` - the sampler will set the ages of all samples in the population to 0.00045 and will not change this value. **Example 2:** `age 0.00032 e` - the sampler will set the ages of all samples in the population to 0.00032 and will treat the age as a free parameter to be estimated from data.

ANCESTRAL-POPS

The ANCESTRAL-POPS module specifies the structure of the population phylogeny, as well as parameters associated with the ancestral populations in the model. Like the CURRENT-POPS module, the ANCESTRAL-POPS module consists of a list of POP sub-modules – one for each of the ancestral populations. Attributes that can be set in the POP sub-module are listed below.

```
ANCESTRAL-POPS-START

        POP-START
                name            AB
                children        A       B
                tau-initial     0.000005
                tau-beta        20000.0
                finetune-tau    0.0000008
        POP-END

        POP-START
                name            ABC
                children        AB      C
                tau-initial     0.00001
                tau-beta        20000.0
                finetune-tau    0.0000008
        POP-END

ANCESTRAL-POPS-END
```

An example of a basic ANCESTRAL-POPS module

POP attribute list (for ancestral population):

name	Name of the ancestral population (50 characters max) **Obligatory**
children	Names of exactly two daughter populations. The populations should be specified 'bottom-up', such that both children are always given before the parent. **Obligatory**
theta-alpha	See current pop specification
theta-beta	See current pop specification
theta-print	See current pop specification
tau-alpha and **tau-beta**	α and β parameter for the Gamma distribution used for prior of the τ parameter associated with the ancestral population **Optional**, default: global values set for all τ and θ parameters in the GENERAL-INFO module
tau-print	A multiplicative factor to use when printing τ parameter to trace file **Optional**, default: global value in the GENERAL-INFO module
tau-initial	An initial value around which to start the MCMC; the initial value is sampled uniformly in the interval [0.8,1.2] X tau-initial **Optional**, default: the prior mean for this τ parameter
finetune-tau	Finetune parameter for the MCMC update of the τ parameter **Optional**, default: the global finetune specified in the GENERAL-INFO module, or dynamically chosen, if find-finetunes is set to TRUE.

MIG-BANDS

The MIG-BANDS module is the only one that can be empty, or missing. It specifies the migration bands to be used in the analysis. The MIG-BANDS module consists of a list of BAND sub-modules – one for each migration band. Attributes that can be set in the BAND sub-module are listed below. Some tips for working with migration bands are given in Section 5 of the paper supplement.

```
MIG-BANDS-START
        BAND-START
                source  C
                target  B
                mig-rate-print 0.1
        BAND-END

        BAND-START
                source  AB
                target  C
                mig-rate-print 0.1
        BAND-END

MIG-BANDS-END
```

An example of a basic MIG-BANDS module

BAND attribute list:

source	Name of the source population of migration (forward in time) **Obligatory**
target	Name of the target population of migration (forward in time) **Obligatory**
mig-rate-alpha and **mig-rate-beta**	α and β parameter for the Gamma distribution used for prior of the rate parameter associated with the migration band **Optional**, default: global values set for all migration rates in the GENERAL-INFO module
mig-rate-print	A multiplicative factor to use when printing migration rates to trace file **Optional**, default: global value in the GENERAL-INFO module

6.1 Data size restrictions

There are no strict restrictions on the number of populations or samples used in analysis. The restrictions are typically implied by the computational resources (time of computation and memory requirements). However, in order to simplify some of the memory allocation procedures in *G-PhoCS*, we implemented **global upper bounds on the number of populations, migration bands and samples**. These upper bounds are represented by constants defined at the top of the patch.c source file (`#define NSPECIES 20, #define MAX_MIG_BANDS 40, #define NS 200` respectively). If your analysis requires larger numbers (say, more than 40 migration bands), you can modify these settings at the top of the patch.c source file, and recompile the source code. Another restriction we apply is for the maximum number of migration events per sampled genealogy (`#define MAX_MIGS 10`). If your analysis requires sampling more migration events, this bound can be modified as well. You will get an indication for this, if you get a "`Fatal Error 0012`" error message in the middle of your run.

7. Using multiple control files

It is possible to specify **primary** and **secondary** control files in the *G-PhoCS* command line. If two files are specified in the command line, the first is treated as the primary file, and the second as the secondary file. This feature is helpful when setting up several alternative runs on the same data. In such cases, a single primary control file can be used, with several secondary files describing alternative settings. The secondary file can contain either module `GENERAL-INFO` or module `MIG-BANDS`, or both, but it cannot contain modules `CURRENT-POPS` and `ANCESTRAL-POPS`. If a `MIG-BANDS` module is given in the second file, the `MIG-BANDS` module of the first file will be completely overridden. However, if a `GENERAL-INFO` module is given in the second file, the specific attributes given there override values set in the first file.

8. Tuning the MCMC

Model priors

In order to obtain robust estimates for the demographic parameters, it is important to explore different settings for the prior distribution. A Gamma distribution is used for all model parameters (except for the locus-specific mutation rate, which is governed by a Dirichlet prior). It is generally recommended to use priors with large variance, so that the prior does not strongly influence the posterior sample. However, more informative priors can be used to expedite convergence of preliminary runs. Parameters priors can be set globally, using the appropriate attributes in the `GENERAL-INFO` module (`mig-rate-alpha`, `mig-rate-beta`, `tau-theta-alpha` and `tau-theta-beta`). In addition, each model parameter can have its prior set individually in the appropriate place (τ and θ in the corresponding `POP` sub-module, and migration rate in the corresponding `BAND` sub-module).

Initial parameter values

The MCMC starts by selecting initial values for all model parameters. By default, values are chosen uniformly in the interval $[0.8,1.2]x m$, where m is the prior mean for the parameter. Due to constraints of the rubber-band operation applied in the updates of parameters for population divergence times, the convergence of these parameters can be slow. Therefore, *G-PhoCS* allows the user to manually set the initialization point for population divergence time parameters. This is done using the `tau-initial` attribute of the corresponding `POP` module.

Finetuning the MCMC updates

Most update steps of the MCMC are governed by finetune parameters. These parameters control the amount of change allowed in that update step. A large finetune will allow large steps that may result in low acceptance rates. Small finetunes are more conservative and lead to higher acceptance rates, but the smaller steps slow down convergence and mixing. Therefore, it is typically recommended to tune the updates to get intermediate acceptance rates (20–70%). The finetune parameters can be set manually by using the appropriate attributes in the control file. Note that *G-PhoCS* has a separate finetune parameter for each of the population divergence time parameters (τ). This feature is useful, since these parameters tend to converge the slowest.

Another feature of ***G-PhoCS*** (introduced in version 1.2) is automatic setting for all finetune parameters. When this feature is turned on (by setting the `find-finetunes` attribute to `TRUE`), the MCMC uses the first X iterations to finetune the acceptance rates of all updates. This is done by searching for finetunes that will result in acceptance rates between 30–40%, through binary search. The user can specify the number of binary search steps (through the `find-finetunes-num-step` attribute), and the number of MCMC iterations in each step (through the `find-finetunes-samples-per-step` attribute). The number of iterations this automatic search takes is the product of these two numbers, and samples taken from these iterations should be discarded and treated as burn-in. The binary search starts with the values supplied in the control file (if such are supplied), and it assumes finetune parameters in the interval (0,10]. If a finetune of 10 results in acceptance rates higher than 40%, this is the finetune that will be used. The status log describes the updates made to the finetune parameters.

```
*****************************************************************
*                  G-Phocs ver 1.3     Jan 2017                 *
*****************************************************************
Setting Maximum Thread Count to: 1
Reading control settings from file sample-control-file.ctl...
Done.
Reading sequence data...  1000 loci, as specified in sequence file seqs-sample.txt.
Reading loci (.=100 loci): ..........

Starting MCMC: 0 burnin, 5000 running, sampled every 0 iteration(s).
There are 11 parameters in the model.
Samples   CoalTimes MigTimes  SPRs      Thetas    MigRates TAU_ 4    TAU_ 5    TAU_ 6    RbberBnd  MutRates  Mixing    | DATA-ln-ld |  TIME
-------------------------------------------------------------------------------------------------------------------------------------------------
   ---  Dynamically finding finetune settings for the first 10000 samples, updating finetunes every 100 samples  ----
-------------------------------------------------------------------------------------------------------------------------------------------------
    100    64.8%     98.5%     72.6%     63.9%     96.0%     19.8%     15.8%     28.7%       0.3%      0.0%     66.3%    |-1348.755834|  0:09
         5.0050000 5.1500000           5.0200000 5.0100000 0.5000000 0.0000004 0.0000004 0.0000014           0.5000000 5.0015000
    200    64.9%     98.8%     70.6%      0.3%     40.6%     24.8%     19.8%     52.5%       0.0%      0.0%      0.0%    |-1348.927122|  0:17
         7.5025000 7.5750000           2.5300000 7.5050000 0.2500000 0.0000002 0.0000002 0.0000021           0.2500000 2.5022500
    300    64.7%    100.0%     70.5%      0.4%     35.6%     39.6%     34.7%     29.7%       0.0%      0.0%      0.0%    |-1348.894125|  0:24
         8.7512500 8.7875000           1.2850000 7.5050000 0.1250000 0.0000002 0.0000002 0.0000018           0.1250000 1.2526250
    400    64.8%    100.0%     70.6%      1.0%     45.5%     52.5%     38.6%     38.6%       0.0%      0.0%      0.0%    |-1349.106040|  0:32
         9.3756250 9.3937500           0.6625000 8.7525000 0.0625000 0.0000003 0.0000002 0.0000018           0.0625000 0.6278125
    500    64.2%    100.0%     70.7%      6.2%     42.6%     16.8%     19.8%     35.6%       0.0%      0.0%      0.0%    |-1348.886927|  0:39
         9.6878125 9.6968750           0.3512500 9.3762500 0.0312500 0.0000002 0.0000001 0.0000018           0.0312500 0.3154062
    600    64.3%      0.0%     70.7%     15.1%     45.5%     18.8%     30.7%     42.6%       0.0%      0.0%      1.0%    |-1348.892750|  0:46
         9.8439062 9.5453125           0.1956250 9.6881250 0.0156250 0.0000002 0.0000001 0.0000020           0.0156250 0.1592031
    700    64.2%    100.0%     70.5%     29.6%     32.7%     21.8%     39.6%     34.7%       0.0%      0.0%      3.0%    |-1348.918140|  0:54
         9.9219531 9.6210938           0.1178125 9.6881250 0.0078125 0.0000001 0.0000001 0.0000020           0.0078125 0.0811016
    800    64.7%    100.0%     70.5%     39.2%     42.6%     58.4%     58.4%     18.8%       0.0%      0.0%     19.8%    |-1348.625812|  1:01
         9.9609766 9.6589844           0.1178125 9.8440625 0.0039062 0.0000002 0.0000001 0.0000019           0.0039062 0.0420508
    900    64.7%    100.0%     70.5%     38.6%     39.6%     51.5%     70.3%     26.7%       0.0%      0.0%     55.4%    |-1348.714501|  1:09
         9.9804883 9.6779297           0.1178125 9.8440625 0.0019531 0.0000003 0.0000003 0.0000014           0.0019531 0.0615762
   1000    64.6%    100.0%     70.3%     40.9%     30.7%     57.4%     61.4%     43.6%       0.0%      0.0%     29.7%    |-1349.013078|  1:16
         9.9902441 9.6874023           0.1567187 9.8440625 0.0009766 0.0000005 0.0000003 0.0000016           0.0009766 0.0518135
```

An example log for the sample run with automatic finetuning

Note: We highly recommend users to use the automatic tuning feature in preliminary runs, as a guide for finding good finetune parameters. However, do note that automatic tuning should not be blindly trusted and used. Acceptance rates may change as the MCMC explores different regions in parameter space, and rates may thus escape the 30%-40% intrerval reached during the tuning iterations.

9. Multi thread support

As of version 1.3, ***G-PhoCS*** can be run using multiple concurrent threads to expedite analysis. Experiments with various data sets indicate a near linear speed-up. For instance, running the program with 4 threads on a 4-core CPU leads to a 3x-3.6x speed up. The multi-threaded program is implemented using **OpenMP**.

Compiling the multi-threaded version:

1. If you compile ***G-PhoCS*** as is, it will compile without multi-threading capabilities and without the need for OpenMP.
2. First, check that OpenMP is installed on your machine. This can be done using the following command pipe in Ubuntu:

   ```
   $ echo |cpp -fopenmp -dM | grep -i openmp
   ```

   If you have OpenMP installed, then the command pipe will output something like

   ```
   #define _OPENMP 201511
   ```

   If the command prints out nothing, then OpenMP is probably not sintalled on your machine.
3. Modify Makefile, by un-commenting the variable assignment on Line 7:

   ```
   # Multi threading enabling flag
   #ENABLE_OMP_THREADS = 1
   ```

   To enable compilation of multi-threaded version, remove the # sign to un-comment line. To compile standard serial version, add the # sign in the beginning of line. Do not change value of variable.

OpenMP is configured at this point. Run "`make`" command to obtain the executables.

Running the multi-threaded version:

Indicate in the command line, how many threads you intent to utilize, with "**`-n NUM`**" option.

```
$bin/G-PhoCS sample-control-file.ctl -n 11
 ***************************************************************
 *                  G-Phocs ver 1.3     Jan 2017                 *
 ***************************************************************
 Setting Maximum Thread Count to: 11
 Reading control settings from file sample-control-file.ctl...
 ...
```

- If you do not specify the number of threads, then OpenMP chooses the number of threads based on the number of cores in your CPU and hyperthreading capabilities.
- This system default cannot be exceeded, even if you specify a larger number of threads in the command line with the **`-n`** option.

10. Trace Analysis

We encourage users to use ***Tracer*** (from the ***BEAST*** package) to view MCMC traces on the fly to evaluate convergence. *readTrace* is a simple program supplied with ***G-PhoCS***, that helps perform simple analyses of the MCMC trace. The readTrace binary is created in the `bin/` subdirectory when you invoke `make`.

```
Usage: bin/readTrace <trace-file-name> [options]
-b, --block-size  SIZE     Blocksize
-d, --discard     NUMBER   Number of samples to discard from beginning of file
-h, --help                 This help page
```

For example, when invoking `bin/readTrace -b500 -d1000`, readTrace discards of the first 1,000 MCMC lines in the trace file, and then writes a line for every 500 trace lines, with the average of the sampled model parameters across these lines. The last block might be smaller than the block size indicated. If the block size is greater than the number of trace lines (or if no block size is given), then a single block is used. Simple scripting languages, such as `awk`, can be used to show a summary for a subset of the model parameters.

`bin/readTrace mcmc.log -d1000 -b500 | awk '{print $5, $6, $7, $8, $9, $10, $11 }'` yields the following output:

```
theta_AB theta_ABC theta_root tau_AB tau_ABC tau_root m_D->B
2.100942 3.158390 4.998728 0.043702 0.056219 0.835663 22.687913
1.632652 3.062018 4.914996 0.055938 0.058402 0.847974 11.878063
0.383683 2.689918 5.255840 0.054393 0.054838 0.699094 15.498366
0.329153 2.727465 5.228274 0.057620 0.058265 0.700874 16.517091
0.402760 2.738567 5.226142 0.057706 0.058898 0.748822 19.034865
0.991482 3.021537 5.045256 0.057257 0.058611 0.783434 21.304190
0.601048 2.789202 5.136031 0.059522 0.059800 0.766115 24.415976
0.855895 2.976421 5.014838 0.056684 0.057656 0.824366 18.043875
```

Output of readTrace

11. Errors and Troubleshooting

G-PhoCS makes regulars checks to test the consistency of its data structures. If an inconsistency is found, it exits with an error message. These errors are typically marked by “Fatal Error X”. Please send the exact error message together with a link to the sequence and control files that generated the error to ilan.gronau@idc.ac.il.

12. Control File Generator

The control file generator is a separate Java application that you may use to assist you in constructing a control file for setting up a ***G-PhoCS*** analysis. It is available from version 1.3.

Launching

```
$java -jar ControlFileGenerator/ControlFileGenerator.jar &
```

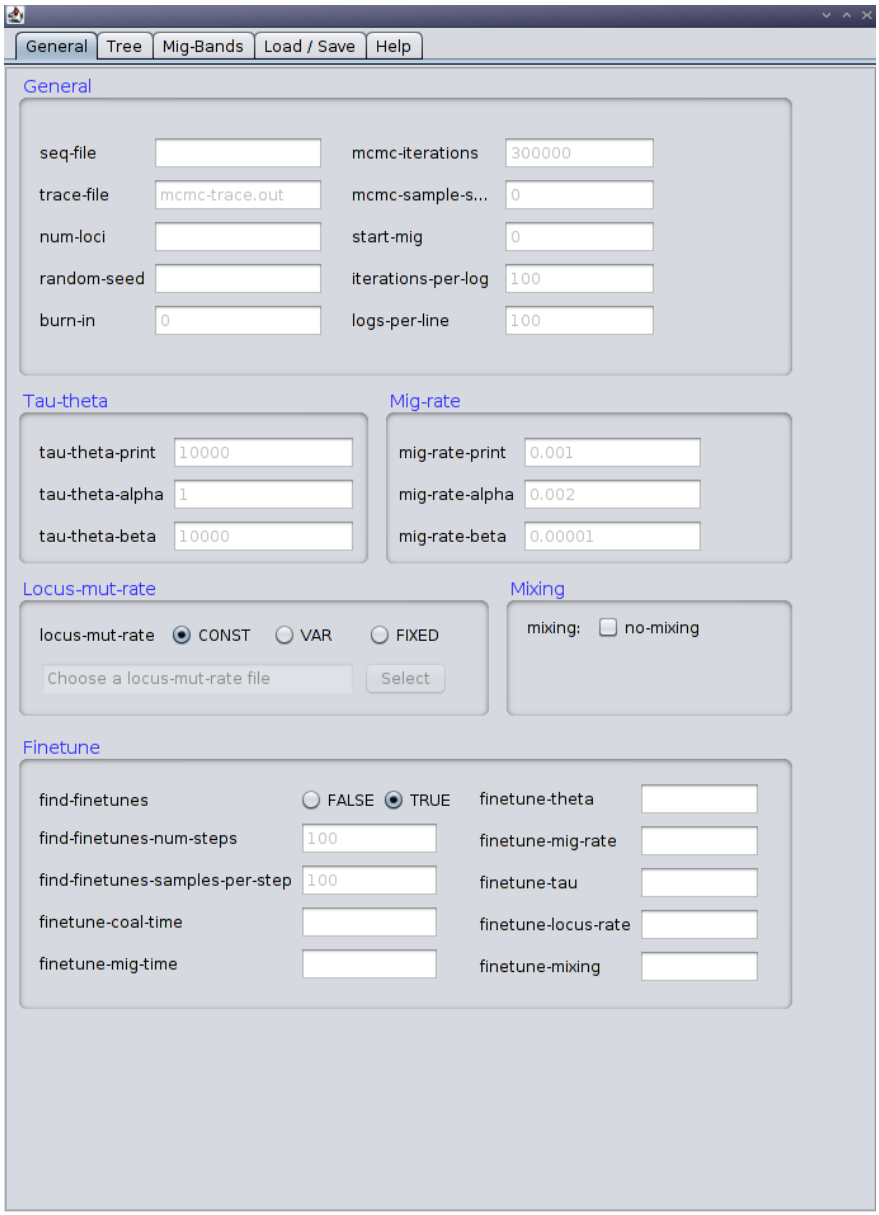
The main window consists of 5 tabs:

- General – for general setup of MCMC algorithm.
- Tree – define the population phylogeny and associate samples with populations
- Mig-bands – set up migration bands between branches of the population phylogeny
- Load/Save – load/save setup from/to a control file
- Help – link to this manual

The ControlFileGenerator is written in Java and runs under JRE version 7 and up If you don't have it installed on your computer, you may download it from https://java.com/en/download/

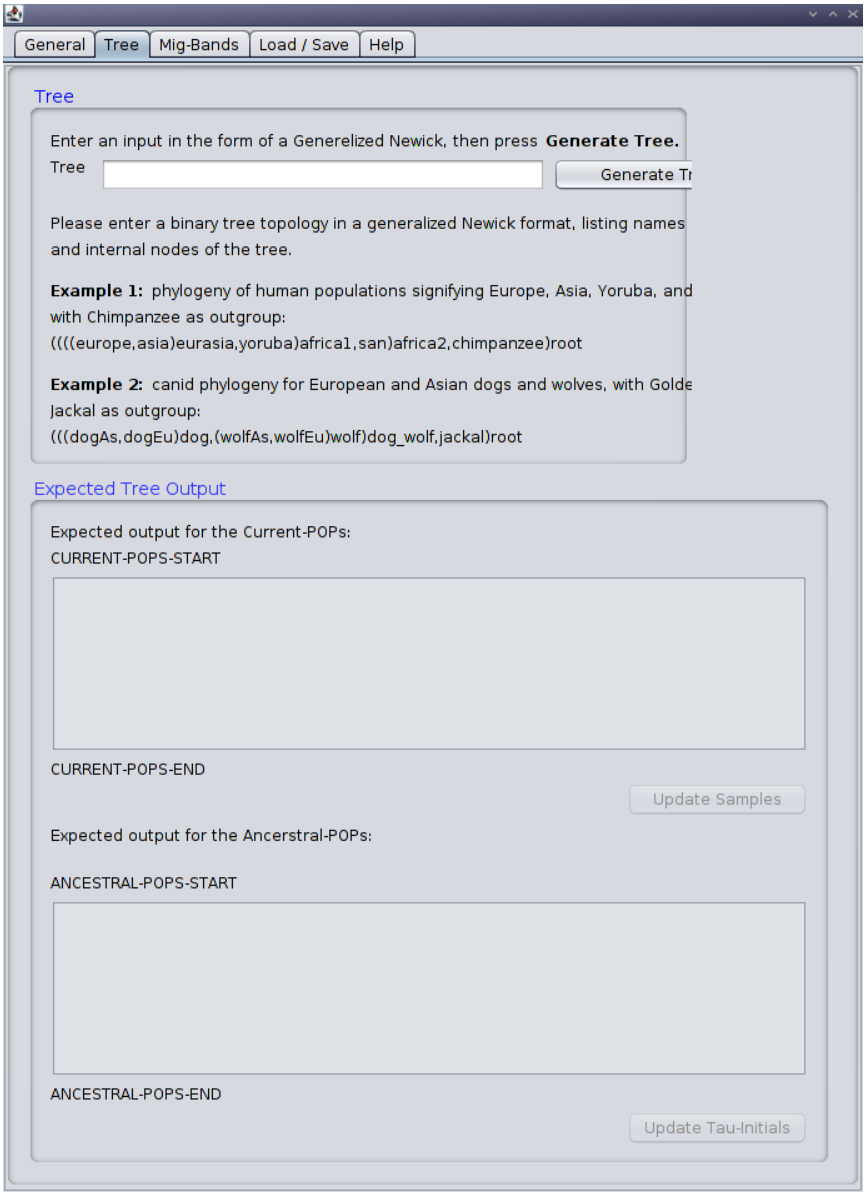
Step 1: General configuration settings

- Specify sequence file
- Specify output and print out parameters
- Set up prior distributions
- Constant / variable mutation rates across loci
- Specify finetune parameters for different update steps, or automatic finetuning (recommended)



Step 2: Tree definition

- Specify population phylogeny using a generalized Newick format (with names for leaves and internal nodes)
- Specify a list of samples associated with each current population (leaves of phylogeny)
- Specify initial values for divergence tmes associated with ancestral populations



Step 3: Migration bands

- Add migration bands to the demographic model by selecting directed pairs of populations.