September 7, 2017

SelEstim version 1.1.7

User Manual

Contents

1 Overview

The software package SelEstim is aimed at distinguishing neutral from selected polymorphisms and estimate the intensity of selection at the latter. SelEstim model accounts explicitly for positive selection, and it is assumed that all marker loci in the dataset are responding to selection, to some extent. SelEstim is written in C. The source code as well as executables for various platforms (currently OS X, Windows, Linux) are available. The C executable reads a data file supplied by the user, and a number of options can be passed through the command line. This document provides information about how to format the data file, how to specify the user-defined parameters, and how to interpret the results.

2 Before you start

2.1 How to get SelEstim?

Download the archive from `http://www1.montpellier.inra.fr/CBGP/software/selestim/`, and extract it from a terminal:

```
unzip SelEstim_1.1.7.zip
```

Binaries for OS X, Windows, Linux are no longer available. Therefore, you need to recompile SelEstim from the source files provided (see the next subsection).

2.2 How to compile SelEstim?

The source files are to be found in the `src` subdirectory. SelEstim is coded using C programming language and can therefore be compiled for any system supported by gcc. To do so, Windows users may need to get a gcc compiler, e.g. by installing MinGW, mingw-64, or Cygwin. To compile the code and get the `selestim` binary, use the Makefile provided:

```
make clean all
```

SelEstim uses OpenMP to implement multithreading, which allows parallel calculation on on computer systems that have multiple CPUs or CPUs with multiple cores. The gcc version included with OS X may generate executable code that results in runtime error ("Abort trap: 6") when more than one thread is used. In that case, you first need to install a recent version of gcc, following the instructions in `http://hpc.sourceforge.net/`.

Then, you can recompile SelEstim using (assuming `gcc` has been installed in `/usr/local/`):

```
make clean all CC=/usr/local/bin/gcc
```

3 Underlying principles of SelEstim

3.1 The data

By default, the data consist of individuals collected in a set of n_d demes, and genotyped at L loci. We denote by n_{ij} the total number of genes sampled in the ith deme at the jth locus, out of which x_{ij} have allelic state A. The vector of allele counts in deme i at locus j therefore reads $\mathbf{n}_{ij} \equiv (x_{ij}, n_{ij}-x_{ij})$. Given the frequencies p_{ij} of the reference allele (the "first" allele in the dataset), the conditional distribution of allele counts $\mathbf{n}_{ij}$ in population i at locus j is binomial:

$$\mathcal{L}(p_{ij};\mathbf{n}_{ij}) = \binom{n_{ij}}{x_{ij}} p_{ij}^{x_{ij}} (1-p_{ij})^{n_{ij}-x_{ij}}. \tag{3.1}$$

Since version 1.1.0, SelEstim can handle pooled-population genotyping data, using the `-pool` option. In that case, the vector of read counts in deme i at locus j therefore reads $\mathbf{c}_{ij} \equiv (r_{ij}, c_{ij} - r_{ij})$. Given the frequencies $\left(\frac{x_{ij}}{n_{ij}}\right)$ of the reference allele in the pool, the conditional distribution of read counts $\mathbf{c}_{ij}$ in population i at locus j is binomial (see Günther and Coop, 2013):

$$\mathcal{L}(\frac{x_{ij}}{n_{ij}};\mathbf{c}_{ij}) = \binom{c_{ij}}{r_{ij}} \left(\frac{x_{ij}}{n_{ij}}\right)^{r_{ij}} \left(1-\frac{x_{ij}}{n_{ij}}\right)^{c_{ij}-r_{ij}}. \tag{3.2}$$

3.2 The population genetics model

The method is based on a diffusion approximation for the distribution of allele frequency in a population subdivided in a number of demes that exchange migrants (i.e., an island model, see Wright, 1931). We recommend you read carefully the details of the model in Vitalis *et al.* (2014). In summary, SelEstim is based on an infinite island model where the ith deme consists of N_i diploid individuals, and receives immigrants from the whole population at rate m_i. The scaled migration parameter in the ith deme is defined as $M_i \equiv 4N_i m_i$. Only bi-allelic markers are considered, i.e. only two alleles (denoted by A and a) may occur at a given locus. The frequency of the reference allele in deme i at locus j is denoted by p_{ij}, and the frequency of the reference allele at the jth locus in the whole population is denoted by π_j. Since it is assumed that the population as a whole is made of an infinite number of islands, π_j gives the frequency of the reference allele in the pool of migrant individuals. The prior distribution of π_j is a beta distribution with shape parameters α and β. The parameters α and β may either be fixed, as in Vitalis *et al.* (2014), using the option `-fixed_beta`, in which case their values is set using the options `-beta_a` and `-beta_b`, respectively. However, by default, the parameters α and β are estimated (see Appendix A).

A simple genic model of selection is considered where, at each locus, the reference allele (say, A) provides a selective advantage. The homozygote individuals AA and the heterozygotes Aa have a relative increase of fitness of $1 + s_{ij}$ and $1 + s_{ij}/2$, respectively, as compared to the aa homozygotes. The scaled coefficient of selection in deme i at locus j is defined as $\sigma_{ij} \equiv 2N_i s_{ij}$. The indicator variable κ_{ij} is defined, which takes the value $\kappa_{ij} = 0$ if the reference allele is selected for, and $\kappa_{ij} = 1$ if the alternative allele allele is selected for. The prior distributions for the selection coefficients σ_{ij} (at each locus, in each deme) are modelled hierarchically (see, e.g., Gelman *et al.*, 2004, pp. 124-125). In particular, σ_{ij} has an exponential prior distribution $f(\sigma_{ij}|\delta_j) \sim \exp\left(\delta_j^{-1}\right)$ that depends upon the locus-specific hyperparameter δ_j, which represents the average effect of selection at locus j (over all demes). This hyperparameter δ_j has an exponential prior distribution $f(\delta_j|\lambda) \sim \exp(\lambda^{-1})$ that depends, in turn, upon the hyperparameter λ, which represents the genome-wide effect of selection over all demes and loci.

The hyperparameter λ may be fixed at a given value (using the option `-fixed_lambda`), or have a prior distribution. In that case, as in Vitalis *et al.* (2014), the prior distribution may be exponential (using the option `-lambda_prior exp`) and therefore $f(\lambda) \sim \exp(\Lambda^{-1})$, where Λ is set using the option `-captl_lambda` (by default, $\Lambda = 1.0$). Otherwise, the prior distribution may be inverse gamma (using the option `-lambda_prior invgam`), as is now set by default. Then, $f(\lambda) \sim \text{InvGamma}(\alpha, \beta)$, where the shape parameter α is set using the option `-invgam_shape` (by default, $\alpha = 3.0$) and where the rate parameter β is set using the option `-invgam_rate` (by default, $\beta = 2.0$).

3.3 The framework for statistical inference

The framework for statistical inference from this model consists in a hierarchical Bayesian model (see Gelman *et al.*, 2004), for which the directed acyclic graph (DAG) is shown in Figure 1. SelEstim is based on a componentwise Markov chain Monte Carlo (MCMC) algorithm to sample from the joint posterior distribution of the model parameters. Some parameters

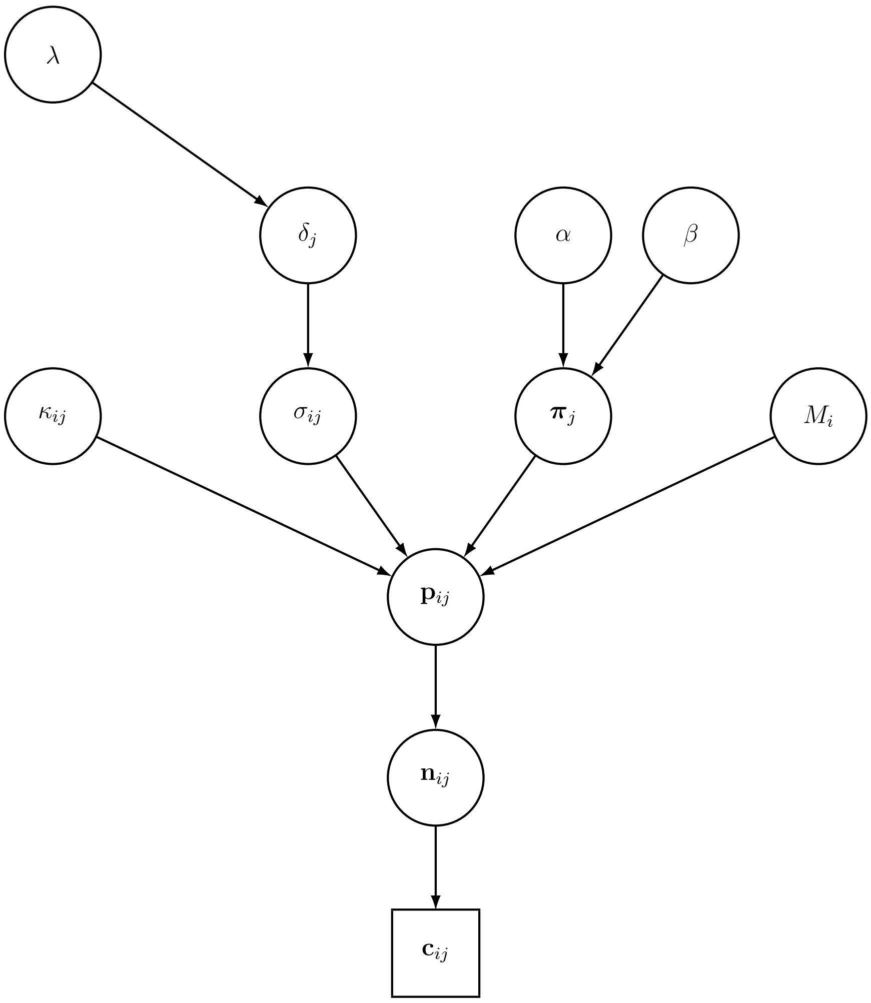


Figure 1: Directed acyclic graph (DAG) of the hierarchical Bayesian model. If the `-pool` option is not set, then the data consist in the allele counts n_{ij} and the read counts $\mathbf{c}_{ij}$ are not considered. If the option `-fixed_beta` is set, then the parameters α and β of the (beta) prior distribution of π_j are not estimated.

of the MCMC algorithm can be adjusted by the user. In particular, proposal distributions are adjusted during short pilot runs, in order to get acceptance rates between 0.25 and 0.40 (see, e.g., Gilks *et al.*, 1996). After the pilot runs, a burn-in period may be defined, before samples are drawn from the Markov chain. Then, samples are taken from the chain, with the number of iterations between any two samples set by the thinning interval. This is aimed at reducing autocorrelations. Typically (as set by default) 100,000 updating steps are completed after 25 short pilot runs of 1,000 iterations each and a burn-in of 50,000 steps. Samples are collected for all the model parameters every 40 steps (thinning), yielding 2,500 observations.

4 Using SelEstim

4.1 Input file format

4.1.1 Allele count data (by default)

The data file reads as follows:

```
--- file begins here ---

6
100
81 19 86 14 2 98 8 92 32 68 23 77
89 11 81 19 9 91 1 99 27 73 27 73
89 11 91 9 11 89 15 85 77 23 80 20

[...97 more lines...]

--- file ends here ---
```

In this example, there are 6 populations (the first number in the file), and 100 SNP markers (the second number in the file). Each line that follows corresponds to one SNP. The number of columns is twice the number of populations. Each pair of numbers corresponds to the allele counts in one population. For example, at the first SNP, in the first population, there are 81 copies of the first allele, and 19 copies of the second allele. In the second population, there are 86 copies of the first allele, and 14 copies of the second allele, etc.

4.1.2 Read count data (using the `-pool` option)

The data file reads as follows:

```
--- file begins here ---

6
100
```

```
50 50 50 50 50 50
71 8 115 0 61 36 51 39 10 91 69 58
82 0 91 0 84 14 24 57 28 80 18 80
93 28 112 30 90 48 0 113 33 68 0 106

[...97 more lines...]

--- file ends here ---
```

In this example, there are 6 populations (the first number in the file), and 100 SNP markers (the second number in the file). The size of each pool (expressed as a number of genes, i.e. twice the number of diploid individuals) is indicated in line 3. In the above example, each pool is made of 50 gene copies (25 diploid individuals). Each line that follows corresponds to one SNP. The number of columns is twice the number of populations. Each pair of numbers corresponds to the allele counts in one population. For example, at the first SNP, in the first population, there are 71 reads of the first allele, and 8 reads of the second allele. In the second population, there are 115 reads of the first allele, and 0 read of the second allele, etc.

4.2 Running SelEstim

SelEstim is a command-line executable. The ASCII hyphen-minus ("-") is used to specify options. As specified below, some options take integer or float values and some options do not. Here is an example call of the program:

```
./selestim -threads 8 -file infile.dat -outputs example -thin 20
           -npilot 5 -burnin 1000 -length 10000
```

In this example run, the data would be read from the file `infile.dat`, and the outputs would be printed out in the `example` subdirectory. 10,000 updating steps would be completed after 5 short pilot runs of 1,000 iterations each and a burn-in of 1,000 steps. Samples would be collected for all the model parameters every 20 steps (thinning), yielding 500 observations. All the options are detailed below, in § 4.7, and the list of output files is provided in § 4.8.

4.3 Sanity checks

4.3.1 Assessing convergence

We advise to assess convergence, e.g., by computing the multivariate extension of Gelman–Rubin's diagnostic (Brooks and Gelman, 1998) on independent Markov chains. The Gelman–Rubin's diagnostic is based on the computation of the ratio of the pooled-chains variance over the within-chain variance. The Gelman–Rubin's diagnostic can be calculated using the coda package (Plummer *et al.*, 2006), as implemented for R (R Core Team, 2013), using the traces of the M_i and λ parameters that are printed out in the `trace_M.out` file and the `trace_lambda.out` file, respectively.

4.3.2 Some convergence issues

We have observed some convergence issues in some particular cases, when the shape parameters α and β of the beta prior distribution of π_j are estimated (option by default). With some datasets, the shape parameters α or β take large values, resulting in a peaked and asymmetric posterior predictive beta distribution for the π_j's. This arises in particular when the reference allele of each SNP is determined based on a reference genome. Since the underlying model of SelEstim ignores such information, we advise to generate datasets where the "first" and the "second" allele at each SNP are chosen randomly. To that end, you may use the `randomize.reference.allele()` function from the `SelEstim.R` file in the `R` subdirectory of the archive.

4.3.3 Checking mixing properties

Also, we strongly recommend assessing the mixing properties of the MCMC by inspecting the trace of the λ parameter in the `trace_lambda.out` file. The trace shall show relatively good mixing (reasonably low autocorrelation, AND no decreasing trend). The autocorrelation can be measured using the `coda` package (Plummer *et al.*, 2006), as implemented for R (R Core Team, 2013). Otherwise, you may want to increase the length of the burn-in period and/or the total length of the Markov chain. Since version 1.1.1, SelEstim also reports the effective sample size (ESS) for various parameters in the `logfile.log` output file. ESS is a measure of how well a Markov chain is mixing. ESS represents the number of effectively independent draws from the posterior distribution that the Markov chain is equivalent to (then ESS must be compared to the chain length). Low ESS (due to strong autocorrelation) indicates poor mixing of the Markov chain. The ESS of the hyper-parameter lambda is typically lower than that of the other parameters.

4.4 Interpreting the results

Because the model assumes that each and every locus in a dataset is selected to a certain extent, one is particularly interested in the posterior densities of the locus-specific hyperparameters δ_j, which are printed out in the file `summary_delta.out`: we expect the density to be shifted toward zero for neutral markers, and to positive values for (presumably) selected loci. Yet, given the hierarchical structure of the model, it would not be sufficient to simply test whether, at a particular locus, the posterior distribution of δ_j departs from zero. This approach would neglect the genome-wide effects of selection. Since it is assumed in the model, that the δ_j's are drawn independently from a common hyperdistribution with parameter λ (that represents the genome-wide effect of selection), it is indeed more appropriate to compare the posterior distributions of the locus-specific coefficients of selection with the "centering" distribution derived from the hyperdistribution of the genome-wide effect of selection. SelEstim uses the Kullback–Leibler divergence (KLD) to measure the distance of the posterior distribution of δ_j from the centering distribution. The KLD estimates for each and every locus are printed out in the file `summary_delta.out`.

In order to provide a decision criterion for discriminating between neutral and selected markers, the KLD is calibrated using simulations from a predictive distribution based on the observed data set. The motivation here is to generate a set of loci equivalent to those that are observed in their levels of diversity and genetic variation. The predictive distribution is parameterized using the estimated posterior means for M_i, κ_{ij}, and λ. If the `-fixed_beta` option is set, then the posterior means for π_j are also used. Otherwise (if the `-fixed_beta` option is not set), then the allele frequencies π_j are drawn from a beta distribution with shape parameters α and β, set to their posterior means. In practice, for each dataset and each analysis, pseudo-observed data (pod) can be generated using either the `-calibration` or the `-calibration_only` option. The pod is then analyzed, using the same MCMC parameters (number and length of pilot runs, burn-in, chain length, etc.) as for the analysis of the original dataset. The KLD values computed for each simulated locus are then combined to obtain an empirical distribution. The quantiles of this empirical distribution are computed, and are used to calibrate the KLD observed for each locus in the original data: e.g., the 99%-quantile of the KLD distribution from the pod analysis provides a 1%-threshold KLD value, which is then used as a decision criterion to discriminate between selection and neutrality. The quantiles are to be found in the `calibration/KLD_quantiles.out` output file.

4.5 R functions

In practice, you may use the R functions from the `SelEstim.R` file in the `R` subdirectory of the archive: the function `plot.delta()` will plot the posterior mean of the δ_j parameter for each locus. You may use the option `plot.delta(map = "my_map.dat")` to sort the markers according to their position. The file `my_map.dat` (the file name is at the user's choice) should contains two columns only: the SNP ID and its position (in bp). Be careful to keep the same order of the SNPs as in the data file.

The function `plot.kld()` will plot the Kullback–Leibler divergence (KLD) for each locus. You may use the option `plot.kld(map = "my_map.dat")` to sort the markers according to their position, as for the function `plot.delta()`. You may also use the `limit` option to compute the threshold value of the empirical distribution of the KLD based on a pod analysis (generated using either the `-calibration` or the `-calibration_only` option); e.g., if you chose `limit = 0.01`, then the 99%-quantile of the KLD distribution from the pod analysis will be used as a decision criterion to discriminate between selection and neutrality. You may also specified the `window.size` and the `n.markers` options if you want to use sliding windows (the width of which is defined by the `window.size` argument). For example, if you use `window.size = 1e6`, 1-Mb windows are constructed for each marker by including all markers that are less than 500 Kb from that focal marker. Outstanding regions are then defined as the windows containing at least `n.markers` SNPs above the critical KLD value (defined from the quantiles of the KLD distribution from the pod analysis).

Last, we also provide the function `compute.F_ST()` in the `SelEstim.R` file, which computes F_{ST} values from the original dataset, either from allele count or from read count data.

4.6 Worked example

In the following, it is assumed that the current (working) directory is at the root of the archive, that contains several files and subdirectories (`data/`, `man/`, `R/`, `src/`). From a terminal, execute the following command line:

```
./src/selestim -file data/data.dat -burnin 5000 -npilot 15
    -lpilot 500 -length 25000 -thin 25 -outputs run-example/
```

In this example run, the data will be read from the file `data/data.dat`, and the outputs will be printed out in the `run-example/` subdirectory. The data consist in a simulation performed with simuPOP, with 4 populations made of 1,000 diploids diverging for 100 generations. A single mutation (at position 4,867,859 bp) is selected for in population 1. To analyse the results, you can load the R functions from the `SelEstim.R` file in the `R/` subdirectory of the archive. Launch R, and set the working directory to the root of the archive, using the `setwd()` function. Then:

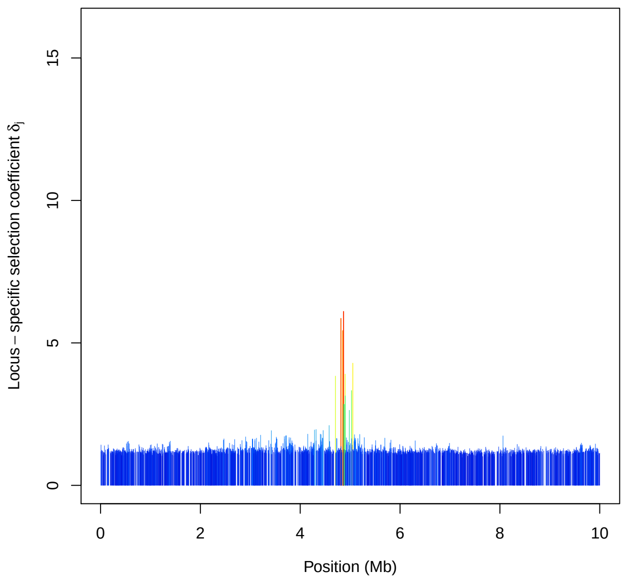
```
> source("R/SelEstim.R")
```

The function `plot.delta()` will plot the posterior mean of the selection parameter for each locus (δ_j). You may use the option `map = "data/data.map"` to sort the markers according to their position. The file `data/data.map` contains two columns only: the SNP ID and its position (in bp). The results can be plotted using:

```
> plot.delta(file = "run-example/summary_delta.out",
    map = "data/data.map")
```

You can also plot the KLD using:

```
> plot.kld(file = "run-example/summary_delta.out",
    map = "data/data.map")
```

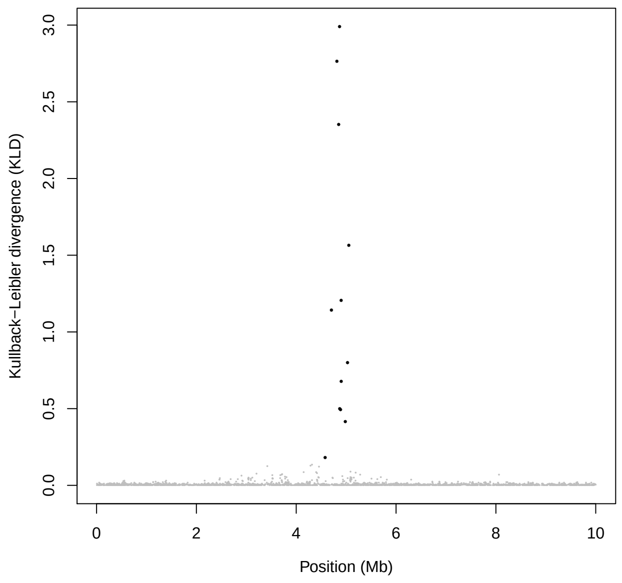


In order to provide a decision criterion for discriminating between neutral and selected markers, the KLD is calibrated using simulations from a predictive distribution based on the observed data set. In practice, pseudo-observed data (pod) can be generated using the `-calibration_only` option of SELESTIM. To do so, execute the following command line from a terminal:

```
./src/selestim -file data/data.dat -burnin 5000 -npilot 15
    -lpilot 500 -length 25000 -thin 25 -outputs run-example/
    -calibration_only -pod_nbr_loci 2000
```

In that example, 2000 markers are simulated for the pod. The KLD values can be plotted using the `plot.kld()` function, and the limit values computed from the pod can be used to provides a decision criterion to discriminate between selection and neutrality. A summary of the limit values is stored in the `calibration/KLD_quantiles.out` output file. The results can be plotted using:

```
> plot.kld(file = "run-example/summary_delta.out",
      map = "data/data.map",calibration_file =
      "run-example/calibration/summary_delta.out",
      limit = 0.001)
```

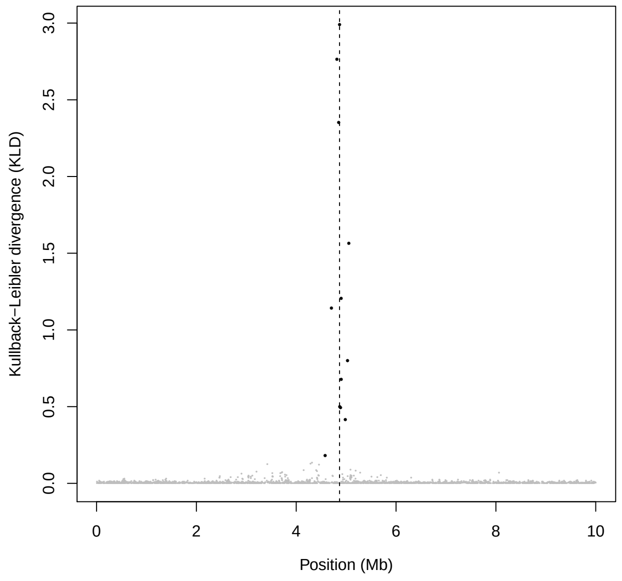


The top SNP (the SNP with the largest KLD) can be identified using:

```
> rslt <- read.table("run-example/summary_delta.out",
      header = TRUE)
> top.snp <- which(rslt$KLD == max(rslt$KLD))
> top.snp
[1] 1124
```

On the previous graph, since the dataset was simulated and we know the truth, one may superimpose the true location of the selected mutation:

```
> plot.kld(file = "run-example/summary_delta.out",
      map = "data/data.map",calibration_file =
      "run-example/calibration/summary_delta.out",
      limit = 0.001)
> abline(v = 4.867859,lty = 2)
```



One can also get the posterior mean of the (scaled) locus-specific selection parameter at that position:

```
> rslt$mean[top.snp]
[1] 6.10151
```

Last, the population-specific scaled coefficient of selection (σ_{ij}) at that postion can be obtained for each deme:

```
> sigma <- read.table("run-example/summary_sigma.out",
      header = TRUE)
> sigma$mean[which(sigma$locus == top.snp)]
[1] 29.972372  5.810732  5.939773  5.759126
```

Here, we conclude that selection is acting in the first deme (which was, actually, the simulated scenario.)

4.7 Details of SelEstim options

`-help`

This option prints out the full list of options accepted by SelEstim i.e.:

```
usage: ./selestim [ options ]
valid options are :
-help  print this message
-version  print version
-file  name of the input file (default: data.dat)
-outputs  directory where the outputs will be produced (default: current directory)
-seed  initial seed for the random number generator (default: computed from current time)
-threads  number of threads to be used (default: number of cpu available)
-length  total length of the Markov chain (default: 100000)
-thin  thinning interval size (default: 40)
-burnin  burn-in length (default: 50000)
-npilot  number of pilot runs (default: 25)
-lpilot  length of each pilot run (default: 500)
-pool  option to analyse data from pooled DNA samples (default: unset)
-fixed_beta  option to fix the shape parameters of the beta prior distribution of pi (default: unset)
-beta_a  shape parameter of the beta prior distribution of pi (default: 0.70)
-beta_b  shape parameter of the beta prior distribution of pi (default: 0.70)
-fixed_lambda  option to fix the value of lambda (default: unset)
-lambda_prior  prior distribution of lambda, which can only be inverse gamma ('invgam', by default) or an exponential ('exp')
-invgam_shape  shape parameter of the inverse gamma prior distribution of lambda (default: 3.00)
-invgam_rate  rate parameter of the inverse gamma prior distribution of lambda (default: 2.00)
-captl_lambda  rate parameter of the exponential prior distribution of lambda (default: 1.00)
-min_M  lower bound for the log-uniform prior on M (default: 0.001)
-max_M  upper bound for the log-uniform prior on M (default: 10000)
-max_sig  upper bound for the exponential prior on sigma (default: 700)
-dlt_cnt  half window width from which updates of allele counts are randomly drawn (default: 5)
-dlt_p  half window width from which updates of p are randomly drawn (default: 0.25)
-dlt_M  standard deviation of the lognormal distribution from which updates of M are drawn (default: 0.10)
-dlt_pi  half window width from which updates of pi are randomly drawn (default: 0.25)
-dlt_sig  standard deviation of the lognormal distribution from which updates of sigma are drawn (default: 2.50)
-dlt_del  standard deviation of the lognormal distribution from which updates of delta are drawn (default: 0.80)
-dlt_lam  standard deviation of the lognormal distribution from which updates of lambda are drawn (default: 0.05)
-dlt_beta_mu  half window width from which updates of the beta mu parameters are drawn (default: 0.03)
-dlt_beta_nu  standard deviation of the lognormal distribution from which updates of the beta nu parameters are drawn (default: 0.50)
-calibration  option to generate pseudo-observed data and calibrate the Kullback-Leibler divergence
-calibration_only  option to generate pseudo-observed data and calibrate the Kullback-Leibler divergence from previous analyses
-pod_nbr_loci  option to specify the number of loci to be simulated for calibration (if different from the dataset)
-verbose  option to print the traces of all parameters (generates big output files!)
```

`-version`

This option prints out the SelEstim version.

`-file`

This option gives the name of the input file. If the option is not specified, the input file name is "data.dat".

`-outputs`

This option gives the directory where all the outputs will be saved. If the option is not specified, then all the output files will be saved in the current directory (where SelEstim is executed).

`-seed`

This option gives the initial seed (integer) for the random number generator. If the option is not specified, then the initial seed is computed from the current computer time. Note that because SelEstim code is parallelized, two different runs with the same initial seed may provide different sequences of random numbers, hence different outputs.

`-threads`

This option gives the number of threads to be used. If the option is not specified, then all available cpu are used.

`-length`

This option gives the total length of the MCMC (i.e., the number of iterations run after the burn-in period). By default, `length` = 100000 (i.e., `-length 100000`).

`-thin`

This option gives the size of the thinning (i.e., the number of iterations between any two records from the MCMC). By default, `thin` = 40 (i.e., `-thin 40`).

`-burnin`

This option gives the length of the burn-in period (i.e., the number of iterations before the first record from the MCMC). By default, `-burnin` = 50000 (i.e., `-burnin 50000`).

`-npilot`

This option gives the number of pilot runs (i.e., the number of runs used to adjust the parameters of the MCMC proposal functions, to get acceptance rates between 0.25 and 0.40). By default, `-npilot` = 25 (i.e., `-npilot 25`).

`-lpilot`

This option gives the length of each pilot run (i.e., the number of iterations for each run). By default, `-lpilot` = 500 (i.e., `-lpilot 500`).

`-pool`

This option enables the analysis of pooled-population genotyping data (see § 3.1). By default, this option is not set.

`-fixed_beta`

This option is used to fix the shape parameters (α and β) of the (beta) prior distribution of π_j. By default, this option is not set.

`-beta_a`

If the `-fixed_beta` option is set, then this option is used to set the shape parameter α of the (beta) prior distribution of π_j. By default, α = 0.7 (i.e., `-beta_a 0.7`).

`-beta_b`

If the `-fixed_beta` option is set, then this option is used to set the shape parameter β of the (beta) prior distribution of π_j. By default, β = 0.7 (i.e., `-beta_b 0.7`).

`-fixed_lambda`

This option is used to fix the value of λ, in which case $\lambda = \Lambda$. By default, this option is not set.

`-lambda_prior`

This option is used to set the prior distribution of λ (see § 3.2). The prior distribution may be an exponential distribution, in which case $f(\lambda) \sim \exp(\Lambda^{-1})$. This is achieved by using the option `-lambda_prior exp` in the command-line. Otherwise, the prior distribution is an inverse gamma, in which case $f(\lambda) \sim \text{InvGamma}(\alpha, \beta)$. This is achieved by using `-lambda_prior invgam` in the command-line (by default).

`-invgam_shape`

If the `-lambda_prior invgam` option is set, then this option is used to set the shape parameter α of the inverse gamma distribution of λ. By default $\alpha = 3.0$ (i.e., `-invgam_shape 3.0`).

`-invgam_rate`

If the `-lambda_prior invgam` option is set, then this option is used to set the shape parameter β of the inverse gamma distribution of λ. By default $\beta = 2.0$ (i.e., `-invgam_rate 2.0`).

`-captl_lambda`

If the `-lambda_prior exp` option is set, then this option gives the value of the hyper-parameter Λ. By default, $\Lambda = 1.0$ (i.e., `-captl_lambda 1.0`).

`-min_M`

This is to avoid computational problems with excessively small M_i values. All moves smaller than `min_M` are discarded (i.e., the chain is kept unchanged). By default, `min_M` = 0.001 (i.e., `-min_M 0.001`).

`-max_M`

This is to avoid computational problems with excessively large M_i values. All moves greater than `max_M` are discarded (i.e., the chain is kept unchanged). By default, `max_M` = 10000 (i.e., `-max_M 10000`).

`-max_sig`

This is to avoid computational problems with excessively large σ_{ij}, δ_j or λ values. All moves greater than `max_sig` are discarded (i.e., the chain is kept unchanged). By default, `max_sig` = 700 (i.e., `-max_sig 700`).

`-dlt_cnt`

This parameter gives the initial value of Δ_n, which is half the window width from which updates of allele counts n'_{ij} are drawn uniformly around the current value n_{ij}. The value of Δ_n is eventually adjusted, for each locus in each deme, during pilot runs to get acceptance rates between 0.25 and 0.40. By default, $\Delta_n = 5$ (i.e., `-dlt_cnt 5`).

`-dlt_p`

This parameter gives the initial value of Δ_p, which is half the window width from which updates of allele frequency p'_{ij} are drawn uniformly around the current value p_{ij}. The value of Δ_p is eventually adjusted, for each locus in each deme, during pilot runs to get acceptance rates between 0.25 and 0.40. By default, $\Delta_p = 0.25$ (i.e., `-dlt_p 0.25`).

`-delt_M`

This option gives the initial value of ν_M, which is the standard deviation on the log scale of the lognormal distribution (with median equal to the current value M_i) from which updates of parameters M_i are drawn. The value of ν_M is eventually adjusted, for each deme, during pilot runs to get acceptance rates between 0.25 and 0.40. By default, $\nu_M = 0.1$ (i.e., `-dlt_M 0.1`).

`-dlt_pi`

This option gives the initial value of Δ_π, which is half the window width from which updates of allele frequency π'_j are drawn uniformly around the current value π_j. The value of Δ_π is eventually adjusted, for each locus, during pilot runs to get acceptance rates between 0.25 and 0.40. By default, $\Delta_\pi = 0.25$ (i.e., `-dlt_pi 0.25`).

`-dlt_sig`

If specified, this option gives the initial value of ν_σ, which is the standard deviation on the log scale of the lognormal distribution (with median equal to the current value σ_{ij}) from which updates of parameters σ'_{ij} are drawn. The value of ν_σ is eventually adjusted, for each locus in each deme, during pilot runs to get acceptance rates between 0.25 and 0.40. If the parameter is not specified, $\nu_\sigma = 2.5$ (i.e., `-dlt_sig 2.5`).

`-dlt_del`

If the `-fixed_lambda` option is not set, and if the `-lambda_prior exp` is set, then this option gives the initial value of ν_δ, which is the standard deviation on the log scale of the lognormal distribution (with median equal to the current value δ_j) from which updates of the hyperparameter δ'_j are drawn. The value of ν_δ is eventually adjusted, for each locus, during pilot runs to get acceptance rates between 0.25 and 0.40. By default, $\nu_\delta = 0.8$ (i.e., `-dlt_del 0.8`).

`-dlt_lam`

This option gives the initial value of ν_λ, which is the standard deviation on the log scale of the lognormal distribution (with median equal to the current value λ) from which updates of the hyperparameter λ' are drawn. The value of ν_λ is eventually adjusted during pilot runs to get acceptance rates between 0.25 and 0.40. By default, $\nu_\lambda = 0.05$ (i.e., `-dlt_lam 0.05`).

`-dlt_mu`

If the `-fixed_beta` option is not set, then the parameters of the (beta) prior distribution of π_j are updated. To that end, we follow Kruschke (2011) and parameterize the beta distribution using $\alpha = \mu\nu$ and $\beta = (1-\mu)\nu$. The `-dlt_mu` option gives the initial value of Δ_μ, which is half the window width from which updates of the mean allele frequency

μ' are drawn uniformly around the current value μ. The value of Δ_μ is eventually adjusted during pilot runs to get acceptance rates between 0.25 and 0.40. By default, $\Delta_\mu = 0.025$ (i.e., `-dlt_mu 0.025`).

`-dlt_nu`

If the `-fixed_beta` option is not set, then the parameters of the (beta) prior distribution of π_j are updated. To that end, we follow Kruschke (2011) and parameterize the beta distribution using $\alpha = \mu\nu$ and $\beta = (1 - \mu)\nu$. The `-dlt_mu` option gives the initial value of ν_ν, which is the standard deviation on the log scale of the lognormal distribution (with median equal to the current value ν) from which updates of the parameter ν' are drawn. The value of ν_ν is eventually adjusted during pilot runs to get acceptance rates between 0.25 and 0.40. By default, $\Delta_\nu = 0.5$ (i.e., `-dlt_nu 0.5`).

`-calibration`

This option is used to generate pseudo-observed data from the posterior predictive distribution, in order to calibrate the Kullback-Leibler divergence measures. When this option is set, a data file containing pseudo-observed data is generated in the `calibration/` subdirectory (this data file is named from the the name of the original dataset, with the prefix `pod_`). A SelEstim analysis is then performed, that results in the same output files as for the original analysis, which are printed out in the `calibration/` subdirectory. An extra file is created, named `calibration/KLD_quantiles.out`, that contains the quantiles of the Kullback–Leibler divergence (KLD) for the pseudo-observed data, which are used for calibration. By default, this option is not set.

`-calibration_only`

This option is used to generate pseudo-observed data from the posterior predictive distribution for an existing analysis, in order to calibrate the Kullback-Leibler divergence measures. It therefore requires that a SelEstim analysis has already been performed. You must indicate in the `outputs` option the directory of the initial analysis. When this option is set, a data file containing pseudo-observed data is generated in the `calibration/` subdirectory (this data file is named from the the name of the original dataset, with the prefix `pod_`). A SelEstim analysis is then performed, that results in the same output files as for the original analysis, which are printed out in the `calibration/` subdirectory. An extra file is created, named `calibration/KLD_quantiles.out`, that contains the quantiles of the Kullback–Leibler divergence (KLD) for the pseudo-observed data, which are used for calibration. By default, this option is not set.

`-pod_nbr_loci`

This option gives the number of loci to be simulated for calibration, if different from the number of loci in the observed data. By default, this option is not set and the pseudo-observed data contain as many loci as the observed data.

`-verbose`

This option is used to print the traces of all parameters in different output files. This may generate very large output files. By default, this option is not set, and only the traces of M_i and λ are printed out.

4.8 Format of the output files

SelEstim produces several output files in the current directory, or in the directory indicated with the `outputs` option; most of these files are also produced in the `calibration/` subdirectory, whenever the options `-calibration` or `-calibration_only` are used:

`logfile.log`

contains all the information that is printed on the console during execution

`diag_mcmc.log`

contains the log(likelihood) along the chain (second column) and the acceptance rates for each category of parameters (up to column 10), i.e.: possibly n_{ij}, p_{ij}, M_i, π_j, possibly the parameters μ and ν of the (beta) prior distribution of π_j, possibly λ, δ_j and σ_{ij}.

`summary_beta.out`

contains the mean and standard deviation (std) of the shape parameters α and β of the (beta) prior distribution of π_j.

`summary_counts.out`

contains the mean and standard deviation (std) of the allele counts x_{ij}, if the `-pool` option is set, for each locus in each deme.

`summary_delta.out`

contains the mean and standard deviation (std) of the δ_j parameter for each locus, as well as the Kullback–Leibler divergence (KLD), that measures the distance of the posterior distribution of δ_j from the "centering" distribution derived from the hyperdistribution with parameter λ distribution.

`summary_freq.out`

contains the mean and standard deviation (std) of the allele frequency p_{ij}, for each locus in each deme.

`summary_kappa.out`

contains the mean the κ_{ij} parameter for each locus in each deme.

`summary_lambda.out`

contains the mean and standard deviation (std) of the λ.

`summary_M.out`

contains the mean and standard deviation (std) of the $M_i \equiv 4N_i m_i$ parameter for each deme.

`summary_pi.out`

contains the mean and standard deviation (std) of the π_j parameter for each locus.

`summary_sigma.out`

contains the mean and the standard deviation (std) of the $\sigma_{ij} \equiv 2N_i s_{ij}$ parameter for each locus in each deme. The the mean and the standard deviation are given unconditional on κ_{ij}, as well as conditional on $\kappa_{ij} = 0$ (first allele selected for) and conditional on $\kappa_{ij} = 1$.

`trace_beta.out`

contains the values of the shape parameters α and β of the (beta) prior distribution of π_j. This might be useful to check for convergence using, e.g., the CODA package in R.

`trace_lambda.out`

contains the value of λ along the chain. This might be useful to check for convergence using, e.g., the CODA package in R.

`trace_M.out`

contains the value of the $M_i \equiv 4N_i m_i$ parameters (in each population) along the chain. This might be useful to check for convergence using, e.g., the CODA package in R.

`trace_counts.out`

contains the value of the allele counts x_{ij} along the chain, if the `-pool` option is set. This file is only printed out if the `-verbose` option is set.

`trace_freq.out`

contains the value of the allele frequency p_{ij} along the chain. This file is only printed out if the `-verbose` option is set.

`trace_pi.out`

contains the value of π_j along the chain. This file is only printed out if the `-verbose` option is set.

`trace_sigma.out`

contains the value of $\sigma_{ij} \equiv 2N_i s_{ij}$ along the chain. This file is only printed out if the `-verbose` option is set.

`trace_delta.out`

contains the value of δ_j along the chain. This file is only printed out if the `-verbose` option is set.

`trace_kappa.out`

contains the value of κ_{ij} along the chain. This file is only printed out if the `-verbose` option is set.

`KLD_quantiles.out`

contains the quantiles of the Kullback–Leibler divergence (KLD) for the pseudo-observed data, which are used for calibration. This file is only printed out (in the `calibration/` subdirectory) if the options `-calibration` or `-calibration_only` are used.

5 Credits

SelEstim uses Makoto Matsumoto and Takuji Nishimura's implementation of the Mersenne Twister random number generator, `http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html`.

6 Copyright

SelEstim is free software under the GNU General Public License (see `http://www.gnu.org/licenses/gpl-3.0.en.html`), and © INRA. The Mersenne Twister code is © 1997 - 2002, Makoto Matsumoto and Takuji Nishimura, and open source code under the BSD Licence.

7 Contact

If you have any question, please feel free to contact me. However, I strongly recommend you read carefully this manual first.

Bibliography

Brooks, S., and A. Gelman, 1998 General methods for monitoring convergence of iterative simulations. J. Comput. Graph. Stat. 7: 434–455.

Gelman, A., J. B. Carlin, H. S. Stern, and D. B. Rubin, 2004 *Bayesian Data Analysis*. Chapman & Hall, New York, 2nd edition.

Gilks, W. R., S. Richardson, and D. J. Spiegelhalter, 1996 *Markov Chain Monte Carlo in Practice*. Chapman & Hall, New York, 2nd edition.

Günther, T., and G. Coop, 2013 Robust identification of local adaptation from allele frequencies. Genetics 195: 205–220.

Kruschke, J. K., 2011 *Doing Bayesian data analysis: A tutorial with R and BUGS*. Academic Press, Oxford.

Plummer, M., N. Best, K. Cowles, and K. Vines, 2006 Coda: output analysis and diagnostics for MCMC. R News 6: 7–11.

R Core Team, 2013 *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Vitalis, R., M. Gautier, K. J. Dawson, and M. A. Beaumont, 2014 Detecting and measuring selection from gene frequency data. Genetics. 196: 799–817.

Wright, S., 1931 Evolution in mendelian populations. Genetics 16: 97–159.

Yang, Z., 2005 Bayesian inference in molecular phylogenetics. In O. Gascuel, editor, *Mathematics of Evolution and Phylogeny*. Oxford University Press, Oxford, 63–90.

Appendix A Details of the new models

New options have been implemented since the published version of the model (Vitalis *et al.*, 2014). In this section, we provide some details on the componentwise Markov chain Monte Carlo algorithm for parameters and/or option that were not detailed in Vitalis *et al.* (2014).

Updating $\mathbf{n}_{ij}$:

If the `-pool` option is set, then the allele counts $\mathbf{n}_{ij} \equiv (x_{ij}, \tilde{n}_i - x_{ij})$ are updated iteratively in each deme, one locus at a time. We assume a uniform prior for x_{ij}, with support from 0 to the size of the pool $\tilde{n}_i$. In the ith deme, at locus j, one allele is chosen at random from a Bernoulli trial with probability 0.5. The new allele count x'_{ij} is chosen as a random variable drawn from a uniform distribution around the current value x_{ij}:

$$x'_{ij} \sim ||U\left(x_{ij} - \Delta_n, x_{ij} + \Delta_n\right)||. \tag{A.1}$$

The size of the interval Δ_n is a constant, which is typically adjusted during 25 short pilot runs of 1,000 iterations, in order to get acceptance rates between 0.25 and 0.40 (see, e.g., Gilks *et al.*, 1996). If x'_{ij} is outside the interval $[0, \tilde{n}_i]$, the excess is reflected back into the interval; that is, if $x'_{ij} < 0$ then x'_{ij} is reset to its absolute value $|x'_{ij}|$, and if $x'_{ij} > \tilde{n}_i$ then x'_{ij} is reset to $2\tilde{n}_i - x'_{ij}$. This proposal is symmetric (Yang, 2005). The updated allele frequency x'_{ij} is therefore accepted according to the appropriate Metropolis probability, which reads:

$$1 \wedge \frac{\mathcal{L}(\frac{x'_{ij}}{\tilde{n}_i}; \mathbf{c}_{ij})\mathcal{L}(p_{ij}; \mathbf{n'}_{ij})}{\mathcal{L}(\frac{x_{ij}}{\tilde{n}_i}; \mathbf{c}_{ij})\mathcal{L}(p_{ij}; \mathbf{n}_{ij})}. \tag{A.2}$$

Equation (A.2) can be rewritten as

$$1 \wedge \frac{\left(\frac{x'_{ij}}{\tilde{n}_i}\right)^{r_{ij}} \left(1 - \frac{x'_{ij}}{\tilde{n}_i}\right)^{c_{ij} - r_{ij}} \binom{\tilde{n}_i}{x'_{ij}} p_{ij}^{x'_{ij}} (1 - p_{ij})^{\tilde{n}_i - x'_{ij}}}{\left(\frac{x_{ij}}{\tilde{n}_i}\right)^{r_{ij}} \left(1 - \frac{x_{ij}}{\tilde{n}_i}\right)^{c_{ij} - r_{ij}} \binom{\tilde{n}_i}{x_{ij}} p_{ij}^{x_{ij}} (1 - p_{ij})^{\tilde{n}_i - x_{ij}}}. \tag{A.3}$$

Updating μ:

If the `-fixed_beta` option is not set, then the parameters of the (beta) prior distribution of π_j are updated. To that end, we follow Kruschke (2011) and parameterize the beta distribution using $\alpha = \mu\nu$ and $\beta = (1 - \mu)\nu$. We assume a uniform prior for μ with support from 0 to 1, i.e. $\mu \sim \mathcal{U}(0, 1)$.

The parameter μ is updated by a random draw from a uniform distribution around the current value μ:

$$\mu \sim U\left(\mu - \Delta_\mu, \mu + \Delta_\mu\right). \tag{A.4}$$

The size of the interval Δ_μ is a constant, which is typically adjusted during 25 short pilot runs of 1,000 iterations, in order to get acceptance rates between 0.25 and 0.40 (see, e.g., Gilks *et al.*, 1996). Since μ is comprised between 0 and 1, if μ' is outside the interval $[0, 1]$, the excess is reflected back into the interval; that is, if $\mu' < 0$ then μ is reset to its absolute value $|\mu'|$, and if $\mu' > 1$ then μ' is reset to $2 - \mu'$. This proposal is symmetric (Yang, 2005). The updated allele frequency μ' is therefore accepted according to the appropriate Metropolis probability, which reads:

$$1 \wedge \frac{\prod_{j=1}^{L} f(\pi'_j)}{\prod_{j=1}^{L} f(\pi_j)}. \tag{A.5}$$

Equation (A.5) can be rewritten as

$$1 \wedge \prod_{j=1}^{L} \left(\frac{\Gamma(\mu\nu)\Gamma((1-\mu)\nu)\pi_j^{\mu'\nu-1}(1-\pi_j)^{(1-\mu')\nu-1}}{\Gamma(\mu'\nu)\Gamma((1-\mu')\nu)\pi_j^{\mu\nu-1}(1-\pi_j)^{(1-\mu)\nu-1}} \right). \tag{A.6}$$

Updating ν:

If the `-fixed_beta` option is not set, then the parameters of the (beta) prior distribution of π_j are updated. To that end, we follow Kruschke (2011) and parameterize the beta distribution using $\alpha = \mu\nu$ and $\beta = (1-\mu)\nu$. We assume an exponential prior for ν, i.e. $\nu \sim \exp(1.0)$. The parameter ν is updated by a random draw from a lognormal distribution with median equal to the current value ν, i.e.:

$$q(\nu \to \nu') = \frac{1}{\nu' s_\nu \sqrt{2\pi}} \exp\left(\frac{-\ln(\nu'/\nu)^2}{2s_\nu^2} \right), \tag{A.7}$$

where s_ν is the standard deviation on the log scale. The standard deviation s_ν is a constant, which is typically adjusted during 25 short pilot runs of 1,000 iterations, in order to get acceptance rates between 0.25 and 0.40. Because the lognormal jumping rule is asymmetric, a Metropolis–Hastings update is required in which the Metropolis ratio is weighted by the ratio of the forward and reverse proposal densities (which is sometimes referred to as the "Hastings term": see, e.g., Gelman *et al.*, 2004, p. 291). This means that when some moves are more likely to happen (because of the asymmetry of the proposal distribution), their probability of acceptance is decreased proportionately. Here, the ratio $q(\nu' \to \nu)/q(\nu \to \nu')$ reduces to ν'/ν. The proposed value ν' is accepted according to the appropriate Metropolis–Hastings probability, which is:

$$1 \wedge \frac{e^{-\nu'} q(\nu' \to \nu) \prod_{j=1}^{L} f(\pi'_j)}{e^{-\nu} q(\nu \to \nu') \prod_{j=1}^{L} f(\pi_j)}. \tag{A.8}$$

Equation (A.8) can be rewritten as

$$1 \wedge \frac{\nu' e^{-\nu'}}{\nu e^{-\nu}} \prod_{j=1}^{L} \left(\frac{\Gamma(\nu')\Gamma(\mu\nu)\Gamma((1-\mu)\nu)\pi_j^{\mu\nu'-1}(1-\pi_j)^{(1-\mu)\nu'-1}}{\Gamma(\nu)\Gamma(\mu\nu')\Gamma((1-\mu)\nu')\pi_j^{\mu\nu-1}(1-\pi_j)^{(1-\mu)\nu-1}} \right). \tag{A.9}$$

Updating λ:

If the `-lambda_prior invgam` option is set, then the parameter λ is updated using Gibbs sampling based on the conditional posterior distribution:

$$f(\lambda|\boldsymbol{\theta}_{[-\boldsymbol{\lambda}]}) \propto \left(\prod_{j=1}^{L} f(\delta_j|\lambda) \right) f(\lambda|\alpha, \beta), \tag{A.10}$$

where $\boldsymbol{\theta}_{[-\boldsymbol{\lambda}]}$ represents all the model parameters but λ. Then:

$$\begin{aligned} f(\lambda|\boldsymbol{\theta}_{[-\boldsymbol{\lambda}]}) &\propto \left(\prod_{j=1}^{L} \frac{1}{\lambda} \exp\left(-\frac{\delta_j}{\lambda}\right) \right) \lambda^{-\alpha-1} \exp\left(-\frac{\beta}{\lambda}\right) \\ &\propto \lambda^{-\alpha-L-1} \exp\left(-\frac{1}{\lambda}\left(\beta + \sum_{j=1}^{L} \delta_j\right)\right) \end{aligned} \tag{A.11}$$

Therefore, the conditional posterior distribution of $\left(\lambda|\boldsymbol{\theta}_{[-\boldsymbol{\lambda}]}\right)$ from equation (A.10) can be rewritten as

$$\left(\lambda|\boldsymbol{\theta}_{[-\boldsymbol{\lambda}]}\right) \sim \text{InvGamma}\left(\alpha + L, \beta + \sum_{j=1}^{L} \delta_j\right). \tag{A.12}$$

Otherwise, if the `-lambda_prior exp` option is set, then the parameter λ is updated from a lognormal distribution with median equal to the current value, as detailed in Vitalis *et al.* (2014).