The Fixed Landscape Inference MethOd (flimo): an alternative to Approximate Bayesian Computation, faster by several orders of magnitude

Sylvain Moinard1*, Edouard Oudet2, Didier Piau3, Eric Coissac1† and Christelle Gonindard-Melodelima1*†

1Univ. Grenoble Alpes, Univ. Savoie Mont Blanc, CNRS, LECA, FR-38000, Grenoble, France.
2Univ. Grenoble-Alpes, CNRS, Laboratoire Jean Kuntzmann (LJK), FR-38000, Grenoble, France.
3Univ. Grenoble-Alpes, CNRS, Institut Fourier, FR-38000, Grenoble, France.

*Corresponding authors. E-mails: sylvain.moinard@univ-grenoble-alpes.fr; eric.coissac@metabarcoding.org; christelle.gonindard@univ-grenoble-alpes.fr;
Contributing authors: edouard.oudet@univ-grenoble-alpes.fr; didier.piau@univ-grenoble-alpes.fr;
†These authors contributed equally to this work.

Abstract

Modelling in biology must adapt to increasingly complex and massive data. The efficiency of the inference algorithms used to estimate model parameters is therefore questioned. Many of these are based on stochastic optimization processes which waste a significant part of the computation time due to their rejection sampling approaches.

We introduce the Fixed Landscape Inference MethOd (flimo), a new
likelihood-free inference method for continuous state-space stochastic models. It applies deterministic gradient-based optimization algorithms to obtain a point estimate of the parameters, minimizing the difference between the data and some simulations according to some prescribed summary statistics. In this sense, it is analogous to Approximate Bayesian Computation (ABC). Like ABC, it can also provide an approximation of the distribution of the parameters.

Two applications are proposed: a usual theoretical example, namely the inference of the parameters of g-and-k distributions; and a population genetics problem, not so simple as it seems, namely the inference of a selective value from time series in a Wright-Fisher model. The results show a drastic reduction of the computational time needed for the inference phase compared to ABC methods, despite an equivalent accuracy. Even when likelihood-based methods are applicable, the simplicity and efficiency of flimo make it a compelling alternative.

The flimo inference method is suitable to many stochastic models involving large data sets. Implementations in Julia and in R are available on https://metabarcoding.org/flimo. To run flimo, the user must simply be able to simulate data according to the chosen model.

Keywords: Approximate Bayesian Computation, Likelihood-free inference, Model optimization, Simulation-based inference, Stochastic models

Introduction

Modelling in biology and ecology presents some important conceptual challenges, due to the increasing complexity and size of the available data. Even for the simplest models, the likelihood is often intractable, especially in population genetics problems [1]. Bayesian rejection-sampling methods [2], where candidate parameters are drawn from a prior distribution, and accepted or not according to some specific criterion, are often used to study such models. In addition to the optimal solution, these methods provide the posterior distribution of the parameters. Among these, Markov chain Monte Carlo (MCMC) methods offer the advantage of yielding convergent estimators of the maximum likelihood of the data [3] but at the cost of some large computational times, and, sometimes, of a complicated preliminary analysis, which is needed to determine the likelihood function of the model. However, to simulate the
studied process for any given set of parameters is often easy. This remark has favored the emergence of methods comparing only some summary statistics of the data, such as moments or quantiles [4], to those of the simulated process. Approximate Bayesian Computation (ABC) methods [5–7] belong to this class of algorithms. They can be improved by being combined with iterative Monte Carlo methods as in Sequential Monte Carlo ABC [8, 9], where the distributions are refined step by step.

Some non-Bayesian methods are also used to estimate the maximum likelihood of a data set with respect to a given model. In this category, one can mention forward algorithms for Hidden Markov Models (HMM) [10], Stochastic Gradient Descent algorithms [11], and Finite Difference Stochastic Approximation methods [12, 13]. While being efficient when the model is adapted, these methods also require in general some significant computation times and a substantial preliminary theoretical analysis.

Like ABC methods, but out of the Bayesian framework, the Fixed Landscape Inference MethOd (flimo), which we propose in the present paper, adjusts some summary statistics of the simulations to those of the data. But the idea behind flimo is to replace the time-consuming rejection-sampling approach by an efficient gradient descent phase. To increase its efficiency, flimo relies on algorithms that are usually only applicable in a deterministic framework. Many efficient local optimization algorithms for deterministic functions [14] exist, such as quasi-Newton algorithms that require Hessian computation. However, these methods need a smooth solution landscape with a limited number of local optima since, when this condition is not met, the convergence cannot be guaranteed. This explains why these methods are not suitable for the optimization of stochastic functions. By definition, for a given set of parameters, a stochastic function may return different values. The non-constancy of
the value returned during different optimization cycles induces an instability of the landscape and, thus, many spurious local optima, preventing the correct estimation of the gradient. To overcome this limitation, in \textit{flimo}, the solution landscape is stabilized by fixing the randomness of the simulation beforehand, drawing all needed random values from a unique random seed uniformly distributed on \([0, 1]\). Later on, these values will be reused for each optimization cycle, by transforming them into the appropriate distribution using quantile functions, a common approach to generate random values with a prescribed distribution. Thus, the simulations become deterministic, and the objective function to be minimized becomes stable. To apply \textit{flimo}, all one needs to do is to simulate the process and to choose some appropriate summary statistics to compare the data and the simulations. The adaptation of existing simulators to \textit{flimo} is thus straightforward. Although \textit{flimo} was developed to provide some point estimates of the unknown parameters, it can also be used to approximate their full distributions.

To illustrate the workings of \textit{flimo}, we present two applications: first, a theoretical benchmark, namely the inference of the parameters of a $g$-and-$k$ distribution from a set of values; second, a population genetics problem, namely the estimation of the allelic selection parameter from some time series in a Wright-Fisher model. In both cases, we compare \textit{flimo} to other existing methods, to highlight the advantages of each approach in terms of bias and precision of estimates as well as computation time.

\textbf{Material and Methods}

\textbf{Description of the algorithm}

The \textit{flimo} algorithm relies on the construction of a regular deterministic objective function which is built from random simulations of the process to
be modelled and which is then efficiently minimized. Contrarily to classical stochastic methods, \textit{flimo} needs a single random draw to evaluate all the candidate parameters $\theta$ and to select the best one according to the chosen summary statistics. This selection is performed using a deterministic algorithm. The steps of the \textit{flimo} algorithm are described below and summarized in Algorithm 1. An illustration is provided in Fig. 1.

\begin{algorithm}
\caption{Fixed Landscape Inference MethOd}
\begin{algorithmic}
\State \textbf{Input:}
\hspace{1em} \#Data
\hspace{2em} $y_{\text{obs}}$
\hspace{1em} \#Defined by model
\hspace{2em} $n_{\text{draw}}$ \#random draws for one simulation
\hspace{2em} $\text{simulatorQ}$ \#adequate quantile simulator
\hspace{1em} \#Chosen by user
\hspace{2em} $n_{\text{sim}}$ \#simulations to perform
\hspace{2em} $s$ \#summary statistics
\hspace{2em} $d$ \#distance between summary statistics
\hspace{2em} $\text{solver}$ \# e.g. optim with IPNewton in Julia
\State \textbf{Output:}
\hspace{1em} $\hat{\theta}$ \# Parameters that minimize $J$
\State \textbf{Ensure:} $\hat{\theta} = \text{argmin} \ J$
\State \#Define quantiles matrix
\State 1: $R \leftarrow (r_{i,j})$ with $r_{i,j} \sim \mathcal{U}([0,1])$ i.i.d. for $1 \leq i \leq n_{\text{sim}}$ and $1 \leq j \leq n_{\text{draw}}$
\State \#Define objective function
\State 2: $J : \theta \mapsto d(s(y_{\text{obs}}), s(\text{simulatorQ}(\theta, R)))$
\State \#Run optimization
\State 3: $\hat{\theta} \leftarrow \text{solver}(J)$
\end{algorithmic}
\end{algorithm}

**Preliminary drawing of the randomness**

A random simulation of a model is based on a fixed number of draws of some simple random variables. For example, in the Wright-Fisher model [15], a binomial distribution is drawn at each generation and another one at each sampling. To run \textit{flimo}, the user needs to determine the number of these draws for a simulation, or at least an upper bound. This value is noted $n_{\text{draw}}$. Then the
user decides the number of simulations $n_{sim}$ to perform to estimate the typical summary statistics of a parameter $\theta$. These simulations can be averaged or used to establish a median for example. This choice is based on a trade-off between computation time and dispersion of the estimators. We suggest to test several values of $n_{sim}$, typically between 10 and 1000. We advise against selecting less than 10 simulations, to avoid the presence of outliers in the inference results. Once $n_{draw}$ and $n_{sim}$ are fixed, the randomness is drawn. A matrix $R$ of dimension $n_{sim} \times n_{draw}$ is set such that each entry $r_{i,j} \sim U([0,1])$ and the entries are independent. These values will be considered as the quantiles of each random draw involved in the $n_{sim}$ simulations and converted on purpose to a realization of the desired random distribution thanks to its quantile function parameterized using $\theta$.

**Special framework to obtain an empirical distribution**

To obtain a convenient empirical distribution, the user must set $n_{sim}$ to 1 and run several independent inferences. To improve performance, it is useful to set the initial condition of inference $n + 1$ to the inferred value of inference $n$.

**Use of the randomness to carry out simulations**

In the chosen model with parameter $\theta$, the $k^{th}$ draw of the process can be written $Z_k \sim L_k(\theta)$ where $L_k$ is a probability distribution parameterized by $\theta$. This draw may depend on the state of the system at step $k - 1$.

The cumulative distribution function (CDF) of $L_k$ is denoted by $F^\theta_k$. Recall that the quantile function $Q^\theta_k$ is defined by $Q^\theta_k(q) = \inf\{x \mid F^\theta_k(x) \geq q\}$ for every $q \in [0,1]$. Thus, $Q^\theta_k$ acts as the inverse function of $F^\theta_k$ since $Q^\theta_k(q) \leq x$ if and only if $q \leq F^\theta_k(x)$. In the most common cases, $F^\theta_k(Q^\theta_k(q)) = q$ for every $q \in [0,1]$. Under $flimo$, each step $Z_k \sim L_k(\theta)$ of simulation $i$ is replaced by equation 1.
\[ Z^i_k = Q^\theta_k(r_{i,k}) \] (1)

By construction, \( Z^i_k \sim \mathcal{L}_k(\theta) \). Indeed, if \( U \sim \mathcal{U}([0,1]) \), then \( Q^\theta_k(U) \) is a random variable with cumulative distribution function \( F^\theta_k \). This method of generation of pseudo-random numbers is called inverse transform sampling. Moreover, once the matrix \( R \) is fixed, each run, and thus the whole set of simulations, becomes deterministic for any value of \( \theta \). This also yields a global monotone coupling since larger position parameters of the distribution yield larger values of the drawn random variables.

Let \( (\theta,R) \mapsto simulatorQ(\theta,R) \) denote the simulator using quantiles instead of random calls. Details about how to write \( simulatorQ \) are given in the Implementation section below. It is crucial to underline that once the matrix \( R \) is fixed, \( simulatorQ(\theta,R) \) is used to produce exactly \( n_{sim} \) independent simulations of the model with respect to \( \theta \), precisely as a classical simulator would do. We note \( y^\theta = simulatorQ(\theta,R) \) the simulations performed for a set of parameters \( \theta \).

**Building the objective function**

The user must then choose some summary statistics \( s \), appropriate for the model studied. Many studies have been carried out on this subject [4]. As for the ABC methods, this choice plays a major role in the quality of inference. One must also choose a distance \( d \) to compare the summary statistics of the data, noted \( y^{obs} \), and the simulations \( y^\theta \). The Euclidean norm and the Mean Absolute Deviation are two reasonable options. Once these components are chosen, the objective function \( J \) is simply defined by equation 2.
The function $J$ is deterministic with the same smoothness as the implied quantile functions, possibly impacted by the choice of the distance $d$ (again for some fixed matrix $R$). The usual continuous probability distributions have a smooth quantile function; on the other hand for discrete distributions (e.g. binomial distributions) the quantile function is piecewise constant.

**Deterministic optimization algorithm and automatic differentiation**

A deterministic local optimization algorithm is then used to estimate $\arg\min_{\theta} J(\theta)$. If the stochastic process involves only draws from continuous distributions (e.g. normal distributions), it is possible to use a gradient-based second-order, e.g. quasi-Newton-type algorithm. In the case of discrete distributions, two routes are available. If $\theta$ is low dimensional, a gradient-free method may be suitable. Otherwise, each discrete distribution can be replaced by an adequate continuous distribution, such as a normal distribution with same mean and variance. The choice of optimization methods is presented in the Implementation section below.

When the (transformed) probability distributions are continuous, the $J$ function is differentiable almost everywhere. It is then possible to accelerate the inference by using an Automatic Differentiation module [16, 17]. Thus, the gradient and the Hessian of $J$ are computed automatically by chain rule and not by finite difference which is the standard method of estimating differentials. Automatic Differentiation reduces both the risks of numerical errors and the
computational times. This makes it possible to process in a short time some problems of relatively high dimension. On the other hand, when used with no precaution, the Automatic Differentiation is “type-unstable” in Julia, which makes its implementation inefficient.

Illustration with a Poisson distribution

An illustration is provided in Fig. 1 and in Supplementary Fig. 1. It shows how \textit{flimo} works and the difference between discrete and continuous distributions. The example is realized in the following framework: \(n_{\text{obs}} = 5\) independent data \((y_{\text{obs}})\) are drawn from a Poisson distribution of parameter \(\theta = 100\), and one wishes to infer the value of \(\theta\). We set \(n_{\text{sim}} = 5\) and \(J\) is the quadratic cost function, defined as in equation 3.

\[
J(\theta) = \left( \frac{1}{n_{\text{obs}}} \sum_{i=1}^{n_{\text{obs}}} y_{i,\text{obs}} - \frac{1}{n_{\text{sim}}} \sum_{i=1}^{n_{\text{sim}}} y_{i,\theta} \right)^2
\]  

(3)

Fig. 1 allows to compare the aspect of the function \(J\), established by usual simulations and in the framework of \textit{flimo}. Supplementary Fig. 1 shows the difference between discrete (here, Poisson) and continuous (here, normal) distributions.

Recall that the quantile function of the Poisson distribution \(\mathcal{P}(\theta)\) is defined by \(Q(r) = k\) if \(e^{-\theta} \sum_{i=0}^{k-1} \frac{\theta^i}{i!} < r \leq e^{-\theta} \sum_{i=0}^{k} \frac{\theta^i}{i!}\). In the continuous approximation, the draws of the Poisson \(\mathcal{P}(\theta)\) simulations are replaced by draws of a normal distribution \(\mathcal{N}(\theta, \theta)\) with the same mean and the same variance, as is customary. Finally, recall that the quantile function of the distribution \(\mathcal{N}(\mu, \sigma^2)\) is \(Q(r) = \mu + \sigma \sqrt{2} \text{erf}^{-1}(2r - 1)\), where \text{erf} denotes the usual error function.
Fig. 1 Illustration of the flimo algorithm. The data set consists of five values from a Poisson distribution from which we want to infer the parameter $\theta$. The average of these values is compared to the average of five simulations. In gray, independent simulations are performed for each value of $\theta$. In blue, the simulations for the different $\theta$ tested are coupled by a single initial random draw, as done by the flimo algorithm.

**Implementation**

**Packages overview**

Implementations of flimo are freely available in the Julia package Jflimo.jl and in the R package flimo, both in [https://metabarcoding.org/flimo](https://metabarcoding.org/flimo). The Julia implementation takes advantage of the very good numerical performances of the Julia language [18]. This implementation is used for both applications presented in the present paper. The R implementation is deposited on the CRAN. While the language R [19] is probably the data analysis language that biologists use the most, it suffers from some performance limitations when compared to Julia. Thus, our R package proposes two modes: flimoR, an
implementation in R of the algorithm, and \texttt{flimoRJ}, which uses the functions implemented in \texttt{Jflimo.jl}.

In Julia, the optimization functions come from the package \texttt{Optim.jl} [20]. The adequate application framework (when the objective function $J$ is differentiable) allows us to use the \texttt{IPNewton} method [21], an interior-point primal-dual Newton algorithm solving nonlinear, constrained optimization problems. One may use this method in combination with the Automatic Differentiation module \texttt{ForwardDiff.jl} [17]. The convergence criterion is usually based on gradient relative variations but it can be chosen by the user. In the case of a one-dimensional non-differentiable problem, the Brent method [22] is adequate. For non-differentiable multidimensional problems, an implementation with the Nelder-Mead optimization method [23, 24] is provided.

In \texttt{flimoR} mode, \texttt{flimo} is applied with the \texttt{optim} function of the R \texttt{stats} package with the L-BFGS-B method [25], a limited-memory modification of the BFGS quasi-Newton method in the differentiable case. In \texttt{flimoRJ} mode, the user must translate the simulation function and the summary statistics function into Julia. The other objects can be written in R. The package \texttt{flimo} encapsulates then directly the package \texttt{Jflimo.jl} with the R package \texttt{JuliaConnectoR} [26].

Building adequate simulator

Any classical simulator of the studied process can be used, simply adapting it according to the following basic procedure. Each time random draws are performed with a random function, this must be replaced by the associated quantile function. To ensure the independence of the draws, each quantile must be used only once for each simulation.
In the R case, one replaces the random functions \texttt{(rpois, rnorm...)} by their quantile version \texttt{(qpois, qnorm...)}. In the Julia case, the procedure is the same with the packages \texttt{Random.jl} and \texttt{Distributions.jl}: the \texttt{rand} calls are replaced by some \texttt{quantile} calls. In both cases, the number of random drawings has to be replaced by the adequate submatrix of \texttt{R}.

**Comparison to other inference algorithms**

We present in detail the two applications used to study \textit{flimo}. One can reproduce our results, using the scripts available on the git page of the project (https://metabarcoding.org/flimo/flimo). All the work was performed on a laptop MacBook Air (2017, 2.2 GHz Intel Core i7 Dual Core Processor).

**Estimate of the parameters of g-and-k distributions**

**Definition**

The family of g-and-k distributions can be viewed as an asymmetric generalization of the normal distributions \( \mathcal{N}(\mu, \sigma^2) \), using two additional shape parameters \( g \) and \( k \). Estimating the parameters of a g-and-k distribution from a random sample is a classical example used to evaluate ABC methods [27]. The density of a general g-and-k distribution is not explicit, but its quantile function is: equation 4 holds for every \( q \in [0, 1] \).

\[
Q(q \mid A, B, g, k) = A + B \left( 1 + 0.8 \frac{1 - \exp(-gz(q))}{1 + \exp(-gz(q))} \right) (1 + z(q)^2)^k z(q) \tag{4}
\]

Here, \( B > 0, k > -1/2, \) and \( z(.) \) is the quantile function of the standard normal distribution. Normal distributions are the \( g = k = 0 \) case of this family since \( Q(\mid A, B, 0, 0) = \mathcal{N}(A, B^2) \).
Simulated data

As is usually done, we choose $\theta_{\text{true}} = (A_{\text{true}}, B_{\text{true}}, g_{\text{true}}, k_{\text{true}}) = (3, 1, 2, 0.5)$ [28]. One generates 100 independent data sets $y^{\text{obs}}$ of 1000 draws of the distribution with parameters $\theta_{\text{true}}$. Supplementary Fig. 2 shows a typical empirical histogram of this distribution.

Compared inference methods

The relative efficiencies of the MCMC, ABC and flimo methods used with different objective functions and computational efforts were measured, our goal being to disentangle the effects of these aspects of the optimization process on the quality of the results (Table 1). Parameter inferences are performed by each method for each of the 100 simulated data sets.

The MCMC method (package winference (available online: https://github.com/pierrejacob/winference) [29]) is considered the gold standard. The Sequential Monte Carlo ABC algorithm [8], used by the package winference and called here $wABC$, uses as summary statistics the Wasserstein distance of order 1, thus comparing the complete collection of empirical quantiles of the two distributions $y^{\text{obs}}$ and $y^\theta$. The objective function $J$ is then defined as an average absolute deviation, see equation 5.

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} |Y_{\text{obs}}^{(i)} - Y_{\theta}^{(i)}|$$  \hspace{1cm} (5)

Here, $n$ denotes the common observation size and the $Y_{(i)}$ are the order statistics $Y_{(1)} \leq \cdots \leq Y_{(n)}$ of the sample $Y = (Y_1, \ldots, Y_n)$ under consideration.

This statistics has been implemented to build the so-called $wflimo$ method. To ease the process, the quantiles are pre-sorted for each simulation so that the realizations are directly order statistics. The objective function based on this Wasserstein distance has several local minima. It was necessary to use the
inferred parameters of iteration $i$ as the initial condition of iteration $i + 1$ and to remove the first four inferences, used as a burn-in phase.

We then explored two strategies to reduce the computation of the ABC and fli:mo methods. The first method aims to reduce the computational effort of the optimization algorithm, allowing it less time to converge (wabc-short method). The second aims to reduce the computational complexity of the objective function by replacing the Wasserstein distance with four summary statistics based on the empirical octiles [28], which significantly reduces the amount of information used (oABC, ofli:mo and ofli:mo-short methods). These summary statistics, called Moment Estimates and denoted by $s(y) = (S_A(y), S_B(y), S_g(y), S_k(y))$ for a set of realizations $y$, characterize the parameters of the distribution. With $E_i$ for $1 \leq i \leq 8$ denoting the empirical octiles of the samples, the summary statistics are defined in equations 6.

\[
\begin{align*}
S_A &= E_4 \\
S_B &= E_6 - E_2 \\
S_g &= \frac{E_6 + E_2 - 2E_4}{E_6 - E_2} \\
S_k &= \frac{E_7 - E_5 + E_3 - E_1}{E_6 - E_2}
\end{align*}
\]  

(6)

Following the recommendations of [30], the simulation procedure is accelerated. Since the summary statistics are based on the empirical octiles of the distribution, it is not necessary to simulate the 1000 realizations of each data set. Indeed, it is possible, using once again the inverse transform simulation method, to simulate uniform distribution order statistics for a data set of size 1000 (the first seven octiles) and to convert them to realizations of a g-and-k distribution with respect to $\theta$. These uniform distribution order statistics
$(U_{(i)})_{1 \leq i \leq 7}$ are simulated with the exponential spacings method [31], using equation 7.

$$V_i \sim \Gamma(1000/8) \text{ independent, } \quad U_{(i)} = \frac{\sum_{j=1}^{i} V_j}{\sum_{k=1}^{7} V_k}$$

Then, the objective function $J$ chosen to evaluate the parameter sets $\theta$ is defined by equation 8.

$$J(\theta) = \left( \frac{S_A(y_{\text{obs}}) - S_A(y^\theta)}{S_A(y_{\text{obs}})} \right)^2 + \left( \frac{S_B(y_{\text{obs}}) - S_B(y^\theta)}{S_B(y_{\text{obs}})} \right)^2 + \left( \frac{S_g(y_{\text{obs}}) - S_g(y^\theta)}{S_g(y_{\text{obs}})} \right)^2 + \left( \frac{S_k(y_{\text{obs}}) - S_k(y^\theta)}{S_k(y_{\text{obs}})} \right)^2$$

| Method          | Implementation        | Summary statistics | Bounds or Prior | Computation time control |
|-----------------|-----------------------|--------------------|-----------------|-------------------------|
| MCMC            | winference :          | -                  | Prior : $U([0, 10]^4)$ | 8000 iterations (burn-in : 2000) |
|                 | metropolis Hastings   |                    | IC : inferred by wABC       |                          |
| oABC            | gk :                  | Moment Estimates   | $n_{sim} = 5 \times 10^6$ |                          |
|                 | 100 best simulations  |                    | $n_{sim} = 10^6$          |                          |
| wABC            | winference :          | 1-Wasserstein      | Prior : $U([0, 10]^4)$    | max time = 180s 1024 particles |
|                 | wsmc                  | distance           | $n_{sim} = 10^6$          |                          |
| wABC-short      |                       |                    | $n_{sim} = 10^6$          |                          |
|                 |                       |                    | $n_{sim} = 10$            |                          |
| oflimo          | Jflimo :              | Moment Estimates   | Bounds : $[0, 10]^4$      | 20 inferences (burn-in : 4) |
| oflimo-short    | IPNewton with AD      |                    | $n_{sim} = 1000$          |                          |
| wflimo          | IPNewton without AD   | 1-Wasserstein      | $n_{sim} = 10$            |                          |
|                 | distance              |                    | $n_{sim} = 10^6$          |                          |

**Table 1** Application framework for 100 inferences of the g-and-k distribution. IC stands for Initial Condition. For each non-flimo method, default setup are used (available online for gk [32] and winference [https://github.com/pierrejacob/winference/blob/master/inst/tutorials/tutorial_gandk.pdf]). Initial Condition for MCMC is the mean value of each parameter inferred by wABC. For wABC inferences, the running time exceeds the time limit set in parameter which is treated as a stop condition if it has been exceeded at the previous iteration.
Estimation of parameters distribution

The \textit{flimo} algorithm can also be used to approximate the distribution of the model parameters, since Bayesian methods provide posterior distributions. The \textit{MCMC} method started with the true parameters is again used as a reference and the obtained posterior marginal distributions are compared to those obtained by \textit{wABC} and \textit{wflimo}. To compare the posterior densities, a Kolmogorov-Smirnov test is used for each of the four marginal distributions of the parameters. This test is overpowered for these sample sizes, so we focused only on the test statistics $D$ and used it as a distance between distributions. Recall that $D$ is a distance of type $L^\infty$ between the CDFs, thus, for CDFs $F_1$ and $F_2$, $D$ is defined by equation 9.

$$
D = \sup_{x \in \mathbb{R}} |F_1(x) - F_2(x)|
$$

Estimate of the selection value in a Wright-Fisher model

Model definition

The \textit{flimo} algorithm has then been applied to infer the strength of selection in a Wright-Fisher model \cite{15} from some time series of allele frequencies \cite{33}. This is a typical study of population genetics problems, with the aim of understanding the molecular mechanisms of selection. The distribution of two alleles $A_0$ and $A_1$ of a locus in a population of size $N_e$ is simulated over several generations, with a selective value $s$ applied on the allele $A_1$. At regular intervals, the allele frequencies are estimated by sampling a part of the population. Noting $X(t)$ the actual proportion of the alternative allele $A_1$ at generation $t$, the model is
simply written with a binomial distribution (equations 10 and 11).

\[
X(t+1) \mid X(t) \sim \frac{1}{N_e} \mathcal{B}(N_e, f(X(t)))
\] (10)

The function \(f\) is defined on \([0, 1]\) by equation 11, where \(s \geq -1\) is the selective value of \(A_1\) and \(h \in [0, 1]\) is the dominance parameter.

\[
f(x) = \frac{x(1 + sh + s(1-h)x)}{1 + 2shx + s(1-2h)x^2}
\] (11)

The available data is sampled at times \(t_1 = 0 < t_2 < \cdots < t_n = T\). Let \(X_k = X(t_k)\). At each observation time, \(n_k\) alleles (\(n_k/2\) individuals) are sampled. The number of \(A_1\) alleles sampled is denoted by \(Y_k\), hence, conditionally on \(X_k\), the distribution of \(Y_k\) is binomial (equation 12).

\[
Y_k \mid X_k \sim \mathcal{B}(n_k, X_k)
\] (12)

**Approach**

We compared the flimo method with the compareHMM method [33]. Thanks to the different approximations implemented in its model, this method is one of the fastest available today, while having a higher accuracy than e.g. the WFABC method [34] used in the same context. CompareHMM is non-Bayesian and relies on Maximum Likelihood Estimation (MLE). It follows a previous approach [10] which uses a forward algorithm on the Hidden Markov Model to compute the likelihood of the model with either the exact binomial model (compareHMM-Bin) or an approximate model using a Beta with spikes (compareHMM-Bws) distribution [35] where transitions from one
generation to the next are represented by a mixture model with a probability that the allele frequency is fixed at 0 or 1, and a Beta distribution conditional on non-fixation otherwise. *CompareHMM-Bin* has an algorithmic complexity of $O(N_e^3)$ so its computation time makes it prohibitive for large populations. It was used as a reference for $N_e = 10^2$ and $N_e = 10^3$. The results of [33] have been reproduced from the python code available online (https://github.com/CyrielParis/compareHMM/). The values of $s$ to be tested are set from $-1$ to $1$ with 300 points with the step used in [33].

For *flimo*, the unknown initial value of $X(0)$ is estimated by $\hat{X}(0) = \frac{Y_0}{n_0}$. The objective to be minimized is the mean absolute deviation around the median, as defined in equation 13.

$$J(\theta) = \text{Mean} \left( |Y^\text{obs}_k - \text{Median}(Y_k(\theta))| \right)_{1 \leq k \leq n + \epsilon |s|}$$  \hspace{1cm} (13)

Here, $\text{Median}(Y_k(\theta))$ is the median of the $n_{\text{sim}}$ simulations done with parameter $\theta$, and $\epsilon = 10^{-2}$. This correction term is present to avoid wrong convergence results because $J$ is almost constant over a wide interval close to the lower bound. We also added a safety: if one reaches a minimum larger than 0.15, another inference is launched to prevent some false convergence due to local minima.

We used three different adaptations of the Wright-Fisher model.

The first model is the Beta with spikes (Bws) approximated model as in *compareHMM-Bws* used with a gradient-based optimization. Hereby, every transition was simulated, contrary to the implementation of [33] which consists in computing by recurrence approximate moments of this distribution.
This model does not work well with \textit{flimo} as the quantiles of the Beta distribution have no closed analytical form. One needs to inverse numerically the CDF $F(x; \alpha, \beta) = I_x(\alpha, \beta)$ with $I_x$ the regularised incomplete beta function, a step which takes a substantial time. The samplings are simulated with approximated normal distributions instead of binomial distributions.

The second model is the Nicholson Gaussian (NG) approximated model \cite{36} with the same optimization process. Classically, each binomial distribution $B(n, p)$ is replaced by the normal distribution $N(np, np(1-p))$, with same mean and variance, with two absorbing states at allele frequencies 0 and 1.

The third model is the original binomial model with a gradient-free optimization. For the gradient-based optimizations, the Automatic Differentiation is not used because it does not speed up the inference in that one-dimensional case. The number of simulations $n_{sim}$ (10 and 200) was chosen for a compromise between efficiency and robustness. The exact model is more difficult to optimize due to its piecewise constant objective function. This is why it leads to worse performances and the approximation of the model is relevant.

\textit{Simulated data}

We considered populations with three effective sizes $N_e = 10^2$, $10^3$, and $10^4$, three $A_1$ selective values $s = 0.01$, 0.1, and 1, and the initial proportion $X(0) = 0.2$. The value $N_e = 10^2$ allows to test the robustness of the inferences despite the strong stochastic variations linked to the small population size (Supplementary Fig. 4), while $N_e = 10^4$ provides a very robust data set. In accordance with a range of tests performed in \cite{33}, we set the dominance parameter to $h = 0.5$ and we simulated over $T = 45$ generations with a sampling every $\Delta t = 5$ generations of $n_k = 0.3N_e$ alleles.
In each scenario, 100 data sets were simulated to compare the performance of the \textit{flimo} method with [33] (Supplementary Fig. 4). The unknown parameters are $X(0)$ and $s$ while the other quantities are assumed to be known.

For the main scenario studied, declined in two effective population sizes $N_e = 10^2$ and $N_e = 10^3$, the selective value is set to $s = 0.1$ (slight selective advantage for $A_1$).

Then we used a single data set with $N_e = 10^4$ and $s = 0.1$ to study the influence of the number of simulations in the dispersion of the estimate, with $n_{\text{sim}}$ chosen in 10, 20, 50, 100, 200, 500 and 1000.

Two other scenarios were also analyzed with $N_e = 10^3$, and with $s = 0.01$ and $s = 1$.

\section*{Results}

\subsection*{Parameter inference for g-and-k distributions}

\subsection*{Point estimate of the parameters}

The \textit{wABC} and \textit{wflimo} methods provide estimates of all four parameters of g-and-k distributions, which are consistent with those obtained by \textit{MCMC}. The averages of the estimates over the 100 simulated data sets yield p-values greater than 0.09 for all the different optimization methods hence, according to Wilcoxon tests with the usual significance level 0.05, they are not statistically distinguishable (Fig. 2a-d).

The accuracy of the three methods is also comparable (Fig. 2a-d). For $A$ and $B$, the variances of the estimates are not significantly different. When comparing \textit{wABC} to \textit{MCMC}, the variance is multiplied by a factor of 1.06 for $A$ and by a factor of 0.99 for $B$ (the p-values of Fisher tests being 0.756 and 0.984, respectively). For the comparison between \textit{wflimo} and \textit{MCMC}, the
increase factors are 1.14 and 1.03, respectively (with p-values 0.522 and 0.898). The shape parameters are estimated with less precision. To wit, the estimates of \( g \) are significantly more dispersed with the \( wABC \) and \( wflimo \) methods than with \( MCMC \), with a variance ratio to \( MCMC \) of 2.38 and 1.97, respectively (p-values < 0.002 for both methods). The dispersion of the estimates of \( k \) is not significantly different between \( wABC \) or \( wflimo \), and \( MCMC \), the respective variance ratio 1.10 and 1.50 has a p-value close to the 5% threshold for the \( wflimo \) versus \( MCMC \) comparison (p-value 0.060). Finally, while no significant difference can be shown between the three methods in terms of bias and accuracy, \( wflimo \) runs 26.4 times faster than \( MCMC \) and 23.6 times faster than \( wABC \) (Fig. 2e).

When the \( wABC \) parameters are adjusted to achieve a computation time comparable to \( wflimo \) (\( wABC \)-short), the method ceases to be reliable and strong biases on the estimates of the parameters \( A \) and \( g \) appear (Fig. 2a and c), as well as an increase in the variance for each parameter compared to \( wABC \) (with variance ratios 2.58 for \( A \), 1.38 for \( B \), 83.7 for \( g \), and 2.39 for \( k \)). If the second strategy, based on a less complex objective function, is applied to the ABC method (\( oABC \)), it allows a correct estimation of the parameters \( A \) and \( g \) but the estimates of \( B \) and \( k \) become strongly biased. It is also less efficient in terms of computation time (Fig. 2e). As regards the effects of these methods of reduction of the computational times, \( flimo \) is more robust. The use of the simplified objective function (\( oflimo \)) reduces the computational time compared to \( wflimo \) by a factor of 5.3 without introducing any bias on the estimates (the p-values comparing the estimates to the simulated parameters are 0.631, 0.139, 0.666, and 0.143). Only the variance of the estimates increases relative to \( wflimo \), by ratios of 1.14, 1.67, 1.84, 4.72 respectively (with p-values 0.5, 0.01, 0.003, < 0.001 respectively). When the computational effort
reduction is applied in conjunction with oflimo (oflimo-short), no significant estimation bias occurs (p-values: 0.803, 0.963, 0.758, 0.497) and no significant increase in the variances of the estimates compared to oflimo is observable (variance ratios: 1.10, 1.10, 1.05, 1.18; p-values: 0.632, 0.622, 0.880, 0.448, see Supplementary Fig. 3). This last method, combining the two optimization procedures, runs 5157 times faster than MCMC, without introducing any significant bias on the parameters estimation and only increasing the variance by a factor of 1.45, and 1.91 for A and B, respectively. In contrast, for the two shape parameters, the increase in variance is much larger: 3.79 and 8.36 for g and k, respectively.

The different versions of flimo were compared for the oflimo-short method. The flimoR version is about 100 times slower than Jflimo.jl (2.3s instead of 0.022s for each inference). The flimoRJ mode has a similar computation time to Jflimo.jl with an additional fixed cost of about 2.5s corresponding to the switch from one language to another, regardless of the number of inferences.

Estimation of the parameters distributions

The quantile function of the density obtained by wflimo is comparable to those obtained by the MCMC and wABC methods. (Fig. 3a-d). For the parameters A and B, the three densities are closely related: the Kolmogorov-Smirnov test statistic D equals 0.076 versus 0.11 for A, 0.082 versus 0.039 for B, for the comparisons wflimo versus MCMC and wABC versus MCMC. For g, the distribution obtained by wflimo is closer to that obtained by MCMC than by wABC (0.25 versus 0.45). For k, the densities obtained by wflimo and wABC are equivalent (0.28 versus 0.21).

However, the current implementation of flimo is not optimized to work in such a mode. Indeed, wflimo takes 252s to perform 1024 inferences, while wABC takes only 192s.
Inference of 100 datasets under g-and-k distribution

Panels a-d: Inference results for each parameter. Medians are plotted as bold black lines. Horizontal dashed lines correspond to the simulated values of the parameters. Panel e: Box plot of the 100 running times in seconds.

Estimate of Wright-Fisher selection values

For $s = 0.1$ and for $N_e = 10^2$ and $N_e = 10^3$, both methods exhibit highly correlated $s$ values over the 100 simulated data sets (namely, $R \in [0.88, 94]$ for $N_e = 10^2$ and $R \in [0.82, 87]$ for $N_e = 10^3$, Fig. 4a-b and Table 2).
The empirical distributions of the four parameters of g-and-k distributions as estimated by different methods. The estimation of the distributions by the method MCMC, used as reference (in grey), is compared to two approximations, the first one provided by the \textit{wABC} method (in blue), the second one by the \textit{wflimo} method (in red).

Compared to \textit{compareHMM-Bin}, for $N_e = 10^2$, \textit{flimo} systematically overestimated $s$ by about $10\% - 15\%$. This systematic over-estimation almost disappears for $N_e = 10^3$, with a difference of estimation between the methods ranging from $0.03\%$ to $3\%$ (Table 2). For $N_e = 10^2$, the inferred mean value of $s$ is 0.883 for \textit{compareHMM-Bin} and 0.896 for \textit{compareHMM-Bws} versus \textit{widehats} $\in \{0.104, 0.114\}$ for the different implementations of \textit{flimo}. For $N_e = 10^3$, we have $\hat{s}_{\textit{compareHMM-Bin}} = 0.0991$, $\hat{s}_{\textit{compareHMM-Bws}} = 0.992$ and $\hat{s}_{\textit{flimo}} \in \{0.1001, 0.1013\}$. This shows that the values inferred by \textit{flimo} are not further from the expected value $s = 0.1$ than the values inferred by \textit{compareHMM}. On average, an inference by \textit{compareHMM-Bin} lasted 1.2s for $N_e = 10^2$ and $1.0 \cdot 10^3$s for $N_e = 10^3$.

For $N_e = 10^4$, a single population was simulated. As expected, the differences between the $s$ values estimated using \textit{flimo} and \textit{compareHMM-Bws} are
very small (less than 1%) even with \( n_{\text{sim}} = 10 \), a very small number of simulations (supplementary Table 1). The only noticeable effect of increasing the number of simulations used by the \textit{flimo} algorithm is to reduce the standard deviation of the estimates by a factor close to \( 10^{-2}/\sqrt{n_{\text{sim}}} \) (Fig. 4c).

| Population size \( N_e \) | Correlation \( 0.9994 \) | 0.89 | 0.91 | 0.93 | 0.93 | 0.88 | 0.94 |
|---------------------------|-------------------|------|------|------|------|------|------|
| \textit{compareHMM}       | \textit{flimo}    | Binomial | Bus | Binomial | Bus | NG |
| \textit{compareHMM-Bws}   | 0.0               | 0.11 | 0.11 | 0.0996 | 0.014 | 0.016 | 0.016 |
| \textit{flimo}            | 1.4               | 0.016 | 0.29 | 0.17 | 3.3 | 0.014 | 0.11 |
| \textit{compareHMM-Bws}   | 0.9997            | 0.85 | 0.87 | 0.82 | 0.88 | 0.84 | 0.87 |
| \textit{flimo}            | 1.4               | 0.037 | 0.69 | 0.34 | 7.1 | 0.013 | 0.10 |

\textbf{Table 2} Inference results, based on 100 simulated data sets, using \textit{compareHMM-Bin}, \textit{compareHMM-Bws} or \textit{flimo} for two different numbers of simulations and three versions of the Wright-Fisher model. Three quantities are presented: the Pearson correlation coefficient, the median of the difference between the values inferred by \textit{compareHMM-Bin} and the other methods, and the median of the computation times.

For the two extreme scenarios \( s = 0.01 \) and \( s = 1 \), \textit{flimo} behaves similarly to what was observed for \( s = 0.1 \) (Supplementary Table 2). For several inferences with \( s = 0.01 \), there is however a systematic overestimation of \textit{flimo} compared to \textit{compareHMM-Bin} (up to a median difference of +32% for the binomial model with 200 simulations). Note that in the latter case, the mean of the estimated \( s \) by \textit{flimo} (\( \hat{s} \) equals 0.0044) is closer to the simulated value \( s = 0.01 \) than with \textit{compareHMM-Bin} (\( \hat{s} = 0.0035 \)). The large deviation from the theoretical value is due to some random fluctuations under these simulation conditions. For \( s = 1 \), the mean values inferred by \textit{compareHMM} are more distant from the true value than those inferred by \textit{flimo} (\( \text{widehats} = 1.02 \) versus \( \text{widehats} \in \{0.996, 1.004\} \)).
Fig. 4  Comparison of inferred selection values $s$ by the \texttt{compareHMM} method and different implementations of \textit{flimo}. \textbf{Panels a and b}: compared inferred values for one hundred data sets with effective population size $N_e = 10^2$ (a) and $N_e = 10^3$ (b). Dashed line corresponds to identity. Black diamond is the simulated value $s = 0.1$. \textbf{Panels c and d}: Influence of the number of simulations on \textit{flimo} inferences for a population of size $N_e = 10^4$. One hundred inferences were run for every number of simulations. \textbf{Panel c}: inferred values of $s$. The dashed line represents the value inferred by \texttt{compareHMM-Bws}, $\hat{s} = 0.098$. \textbf{Panel d}: standard deviation of the inferred values of $s$ with linear regression curve.

Discussion

Concerning the punctual inference of parameters, \textit{flimo} has the advantage of being considerably faster than the other methods with comparable accuracy: up to 5,000 times faster for the g-and-k example, and up to 100 times faster on the Wright-Fisher model. Obviously, the efficiency of an algorithm strongly
depends on its implementation, especially on the programming languages used. If the algorithm \textit{flimo} is implemented in Julia, a language well known for its numerical performances, the other algorithms used here are implemented either in C++ for \textit{MCMC} and \textit{wABC}, or in Python with \texttt{Numpy} for \textit{compareHMM}, hence, in some languages with comparable performances [37]. This ensures that its computation speed is an intrinsic property of the \textit{flimo} algorithm.

Like in ABC methods, the choice of the summary statistics is important. However, \textit{flimo} seems to be less sensitive than the ABC implementations tested here. This lower sensitivity allows to select some summary statistics that can be calculated quickly, thus decreasing the optimization time of \textit{flimo}. This is illustrated by the use of the Wasserstein distance which is necessary to have good estimates of the parameters of g-and-k distributions using the ABC method, while it only slows down \textit{flimo} without any benefit on the accuracy of the results. The same is true for the use of the Beta with spikes distribution used by the \textit{compareHMM} algorithm, and which similarly only slows down \textit{flimo} because of the complexity of its implementation.

Like ABC methods, \textit{flimo} allows to obtain suitable empirical estimates of the distribution of the inferred parameters, but at the cost of its computation efficiency. An interesting strategy could be to implement some hybrid methods, using \textit{flimo} to circumscribe a region of interest in the parameter space, which could then be used as a fine prior for a Bayesian method.

Point estimators are obtained in an extremely efficient way in terms of computation time and with good accuracy. This fact shows that \textit{flimo} is a solid alternative to other inference methods, especially ABC, with a simple implementation which, thanks to the Julia and R packages that we developed, can easily be adapted to other contexts.
By rethinking the role of randomness in stochastic model inference problems, \textit{flimo} allows to use efficient deterministic gradient-based optimization algorithms to infer parameters of probabilistic models whose likelihood or moments are intractable. All these qualities make \textit{flimo} a particularly simple and efficient inference method.

\textbf{Supplementary information.}
### Supplementary Table 1

Inference results, based on a single simulated data set with $N_e = 10^4$, using `compareHMM-Bws` or `flimo` for different numbers of simulations and three versions of the Wright-Fisher model. Two quantities are presented: the median of the difference between the values inferred by the `flimo` and `compareHMM-Bws` methods, and the median of the computation times.
**flimo**

| Selective value | Criterion | \( N_e = 10^3 \) | \( s = 0.01 \) | \( s = 1 \) |
|-----------------|-----------|------------------|----------------|----------------|
| \( \text{Bus} \) | \( \text{Bus} \) | \( \text{Bus} \) | \( \text{NG} \) | \( \text{NG} \) |
| Correlation    | 0.998     | 0.86             | 0.89           | 0.85           | 0.89           | 0.88           | 0.90           |
| Median difference | 0.0       | 3.2 \( 10^{-3} \) | 2.5 \( 10^{-3} \) | -5.0 \( 10^{-4} \) | 2.8 \( 10^{-3} \) | 4.3 \( 10^{-4} \) | 9.5 \( 10^{-4} \) |
| \( \hat{s}_{\text{flimo}} - \hat{s}_{\text{compareHMM-Bus}} \) | (0 %) | (32%) | (25%) | (-5.0%) | (28%) | (4.3%) | (9.5%) |
| Seconds by inference | 1.4 | 0.038 | 0.76 | 0.34 | 6.4 | 0.013 | 0.10 |
| Correlation    | 0.98      | 0.75             | 0.72           | 0.65           | 0.69           | 0.64           | 0.66           |
| Median difference | -6.7 \( 10^{-3} \) | -2.1 \( 10^{-3} \) | -1.5 \( 10^{-3} \) | -3.0 \( 10^{-3} \) | -1.8 \( 10^{-2} \) | -2.5 \( 10^{-2} \) | -2.2 \( 10^{-2} \) |
| \( \hat{s}_{\text{flimo}} - \hat{s}_{\text{compareHMM-Bus}} \) | (-40.67%) | (-2.1%) | (-1.5%) | (-3.0%) | (-1.8%) | (-2.5%) | (-2.2%) |
| Seconds by inference | 3.0 | 0.015 | 0.29 | 0.22 | 3.8 | 0.015 | 0.12 |

**Supplementary Table 2** Inference results, based on 100 simulated data sets with \( N_e = 10^3 \) and \( s \in \{0.01, 1\} \), using \textit{compareHMM} or different implementations of the \textit{flimo} method. Three quantities are presented: the Pearson correlation coefficient, the median of the difference between the values inferred by the \textit{flimo} or \textit{compareHMM-Bus}, and \textit{compareHMM-Bin} methods, and the median of the computation times.
Supplementary Figure 1  Illustration of the flimo algorithm for a Poisson distribution (See Fig. 1). In blue, flimo with the original Poisson distribution. In red, the draws of the Poisson simulations are replaced by draws from a normal distribution with same moments. For the continuous approximation, the draws of the Poisson $P(\theta)$ simulations are replaced by draws from a normal distribution $\mathcal{N}(\theta, \theta)$. 
Supplementary Figure 2  Empirical histogram of a g-and-k distribution.
Supplementary Figure 3  Study of the influence of the number of simulations for *flimo*: standard deviation of the estimated parameters divided by the true parameter value, by the number of simulations used. Both scales are logarithmic.
Supplementary Figure 4 One hundred simulated data sets for $s = 0.1$ and two population sizes (Panel a: $N_e = 10^2$; Panel b: $N_e = 10^3$) for which the selective value was inferred by the flimo and compareHMM methods (Table 2). Thick black line is the average $A_1$ proportion at each sampled time. Panel a: the dispersion from one simulation to another is important with a non-negligible probability of allele fixation.
Acknowledgments. We thank Pierre Pudlo and Adeline Leclercq-Samson for their help on some theoretical aspects of the paper, and Simon Boitard for sharing with us his work on the Wright-Fisher model.

This work was supported by the Alpalga project (ANR-20-CE02-0020).

Declarations

Data Accessibility Statement

All the code used to produce the results of this paper is available on the git page of the project https://metabarcoding.org/flimo.

Authors’ contributions

All authors conceived the algorithm. S.M., C.G. and E.C. wrote the manuscript. E.O. and D.P. contributed to writing the manuscript. S.M. developed the R and Julia packages and performed computational experiments with assistance and guidance from C.G., E.C. and E.O. C.G. and E.C. supervised the project.

Competing interests

The authors declare no competing interests.

References

[1] Stephens M. Inference Under the Coalescent. In: Balding DJ, Bishop M, Cannings C, editors. Handbook of Statistical Genetics. Chichester: John Wiley & Sons, Ltd; 2004. p. bbc22. Available from: https://onlinelibrary.wiley.com/doi/10.1002/0470022620.bbc22.
[2] Shoemaker JS, Painter IS, Weir BS. Bayesian statistics in genetics: a guide for the uninitiated. Trends in Genetics. 1999 Sep;15(9):354–358. https://doi.org/10.1016/S0168-9525(99)01751-5.

[3] Stephens M, Donnelly P. Inference in molecular population genetics. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 2000 Nov;62(4):605–635. https://doi.org/10.1111/1467-9868.00254.

[4] Nunes MA, Balding DJ. On Optimal Selection of Summary Statistics for Approximate Bayesian Computation. Statistical Applications in Genetics and Molecular Biology. 2010 Jan;9(1). https://doi.org/10.2202/1544-6115.1576.

[5] Tavaré S, Balding DJ, Griffiths RC, Donnelly P. Inferring Coalescence Times From DNA Sequence Data. Genetics. 1997 Feb;145(2):505–518. https://doi.org/10.1093/genetics/145.2.505.

[6] Pritchard JK, Seielstad MT, Perez-Lezaun A, Feldman MW. Population growth of human Y chromosomes: a study of Y chromosome microsatellites. Molecular Biology and Evolution. 1999 Dec;16(12):1791–1798. https://doi.org/10.1093/oxfordjournals.molbev.a026091.

[7] Beaumont MA, Zhang W, Balding DJ. Approximate Bayesian Computation in Population Genetics. Genetics. 2002 Dec;162(4):2025–2035. https://doi.org/10.1093/genetics/162.4.2025.

[8] Del Moral P, Doucet A, Jasra A. An adaptive sequential Monte Carlo method for approximate Bayesian computation. Statistics and Computing. 2012 Sep;22(5):1009–1020. https://doi.org/10.1007/s11222-011-9271-y.
[9] Dean TA, Singh SS, Jasra A, Peters GW. Parameter Estimation for Hidden Markov Models with Intractable Likelihoods: Estimating intractable HMMs. Scandinavian Journal of Statistics. 2014 Dec;41(4):970–987. https://doi.org/10.1111/sjos.12077.

[10] Bollback JP, York TL, Nielsen R. Estimation of $2N_e s$ From Temporal Allele Frequency Data. Genetics. 2008 May;179(1):497–502. https://doi.org/10.1534/genetics.107.085019.

[11] Bottou L. Stochastic Gradient Descent Tricks. In: Montavon G, Orr GB, Müller KR, editors. Neural Networks: Tricks of the Trade. vol. 7700. Berlin, Heidelberg: Springer Berlin Heidelberg; 2012. p. 421–436. Series Title: Lecture Notes in Computer Science. Available from: http://link.springer.com/10.1007/978-3-642-35289-8_25.

[12] Kiefer J, Wolfowitz J. Stochastic Estimation of the Maximum of a Regression Function. The Annals of Mathematical Statistics. 1952 Sep;23(3):462–466. https://doi.org/10.1214/aoms/1177729392.

[13] Spall JC. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Transactions on Automatic Control. 1992 Mar;37(3):332–341. https://doi.org/10.1109/9.119632.

[14] Venter G. Review of Optimization Techniques. In: Blockley R, Shyy W, editors. Encyclopedia of Aerospace Engineering. Chichester, UK: John Wiley & Sons, Ltd; 2010. p. eae495. Available from: https://onlinelibrary.wiley.com/doi/10.1002/9780470686652.eae495.

[15] Ewens WJ. Mathematical Population Genetics, 1. Theoretical Introduction. vol. 27 of Interdisciplinary Applied Mathematics. New York, NY:
[16] Bartholomew-Biggs M, Brown S, Christianson B, Dixon L. Automatic differentiation of algorithms. Journal of Computational and Applied Mathematics. 2000 Dec;124(1-2):171–190. https://doi.org/10.1016/S0377-0427(00)00422-2.

[17] Revels J, Lubin M, Papamarkou T. Forward-Mode Automatic Differentiation in Julia. arXiv:160707892 [cs]. 2016 Jul;ArXiv: 1607.07892.

[18] Bezanson J, Edelman A, Karpinski S, Shah VB. Julia: A Fresh Approach to Numerical Computing. SIAM Review. 2017 Jan;59(1):65–98. https://doi.org/10.1137/141000671.

[19] R Core Team. R: A Language and Environment for Statistical Computing. Vienna, Austria; 2021. Available from: https://www.R-project.org/.

[20] K Mogensen P, N Riseth A. Optim: A mathematical optimization package for Julia. Journal of Open Source Software. 2018 Apr;3(24):615. https://doi.org/10.21105/joss.00615.

[21] Wächter A, Biegler LT. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming. 2006 Mar;106(1):25–57. https://doi.org/10.1007/s10107-004-0559-y.

[22] Brent RP. Algorithms for minimization without derivatives. Unabridged republication of the work published by Prentice-Hall, Englewood Cliffs, NJ, 1973 ed. Dover books on Mathematics. Mineola, NY: Dover Publications; 2002.
[23] Nelder JA, Mead R. A Simplex Method for Function Minimization. The Computer Journal. 1965 Jan;7(4):308–313. https://doi.org/10.1093/comjnl/7.4.308.

[24] Gao F, Han L. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. Computational Optimization and Applications. 2012 Jan;51(1):259–277. https://doi.org/10.1007/s10589-010-9329-3.

[25] Byrd RH, Lu P, Nocedal J, Zhu C. A Limited Memory Algorithm for Bound Constrained Optimization. SIAM Journal on Scientific Computing. 1995 Sep;16(5):1190–1208. https://doi.org/10.1137/0916069.

[26] Lenz S, Hackenberg M, Binder H. The JuliaConnectoR : A Functionally-Oriented Interface for Integrating Julia in R. Journal of Statistical Software. 2022;101(6). https://doi.org/10.18637/jss.v101.i06.

[27] Sisson SA, Fan Y, Beaumont MA, editors. Handbook of Approximate Bayesian Computation. 1st ed. Boca Raton, Florida : CRC Press, [2019]: Chapman and Hall/CRC; 2018. Available from: https://www.taylorfrancis.com/books/9781439881514.

[28] Drovandi CC, Pettitt AN. Likelihood-free Bayesian estimation of multivariate quantile distributions. Computational Statistics & Data Analysis. 2011 Sep;55(9):2541–2556. https://doi.org/10.1016/j.csda.2011.03.019.

[29] Bernton E, Jacob PE, Gerber M, Robert CP. Approximate Bayesian computation with the Wasserstein distance. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 2019 Apr;81(2):235–269. https://doi.org/10.1111/rssb.12312.
[30] Fearnhead P, Prangle D. Constructing summary statistics for approximate Bayesian computation: Semi-automatic Approximate Bayesian Computation. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 2012 Jun;74(3):419–474. https://doi.org/10.1111/j.1467-9868.2011.01010.x.

[31] Ripley BD. Stochastic Models. In: Wiley Series in Probability and Statistics. Hoboken, NJ, USA: John Wiley & Sons, Inc.; 1987. p. 96–117. Available from: https://onlinelibrary.wiley.com/doi/10.1002/9780470316726.ch4.

[32] Prangle D. gk: An R Package for the g-and-k and generalised g-and-h Distributions. arXiv:170606889 [stat]. 2017 Jun;ArXiv: 1706.06889.

[33] Paris C, Servin B, Boitard S. Inference of Selection from Genetic Time Series Using Various Parametric Approximations to the Wright-Fisher Model. G3 Genes|Genomes|Genetics. 2019 Dec;9(12):4073–4086. https://doi.org/10.1534/g3.119.400778.

[34] Foll M, Shim H, Jensen JD. WFABC: a Wright-Fisher ABC-based approach for inferring effective population sizes and selection coefficients from time-sampled data. Molecular Ecology Resources. 2015 Jan;15(1):87–98. https://doi.org/10.1111/1755-0998.12280.

[35] Tataru P, Bataillon T, Hobolth A. Inference Under a Wright-Fisher Model Using an Accurate Beta Approximation. Genetics. 2015 Nov;201(3):1133–1141. https://doi.org/10.1534/genetics.115.179606.
[36] Nicholson G, Smith AV, Jonsson F, Gustafsson O, Stefansson K, Donnelly P. Assessing population differentiation and isolation from single-nucleotide polymorphism data. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 2002 Oct;64(4):695–715. https://doi.org/10.1111/1467-9868.00357.

[37] Aruoba SB, Fernández-Villaverde J. A comparison of programming languages in macroeconomics. Journal of Economic Dynamics and Control. 2015 Sep;58:265–273. https://doi.org/10.1016/j.jedc.2015.05.009.