Using `ABCtoolbox`

Daniel Wegmann[*][†], Christoph Leuenberger [‡] and Laurent Excoffier[*]

September 30, 2009

Contents

[*]University of Bern, Computational and Molecular Population Genetics Laboratory, 3012 Bern, Switzerland
[†]To whom correspondence should be addressed, daniel.wegmann@zoo.unibe.ch
[‡]Ecole d'ingénieurs de Fribourg, Bd. de Pérolles 80, 1705 Fribourg, Switzerland

1 An Introduction to `ABCtoolbox`

`ABCtoolbox` was designed to perform Approximate Bayesian Computation (ABC) estimations using various recently published algorithms including MCMC without likelihood and Population Monte Carlo. Due to its potential to interact with almost any command line simulation software, `ABCtoolbox` can be used to study problems in different areas including genomics or population genetics. This first chapter of the manual is designed to give a short introduction into Approximate Bayesian Computations and the use of `ABCtoolbox`. We advise the reader to start with this section. A more detailed (mathematical) description of all methods implemented in `ABCtoolbox` is given in Section 6 "ABC – the Methodological Reference" on page 54.

1.1 A very short Introduction to ABC

1.1.1 Bayesian Statistics

With the advent of ever more powerful computers and the refinement of algorithms like Markov Chain Monte Carlo (MCMC) or Gibbs sampling, Bayesian statistics has become an important tool for scientific inference during the past two decades, especially in population genetics (Marjoram and Tavare 2006). Using examples from this field of science we will outline here the basic principles of Bayesian inference in general, and Approximate Bayesian Computations in particular. Note, however, that the methods are equally well applicable to any other model or question.

Consider a model $\mathcal{M}$ generating data $\mathcal{D}$ (DNA sequence data, for example) determined by parameters $\boldsymbol{\theta}$ whose joint prior density we denote by $\pi(\boldsymbol{\theta})$. The quantity of interest is the posterior distribution of the parameters which can be calculated by the Bayesian rule

$$\pi(\boldsymbol{\theta}|\mathcal{D}) = c \cdot f_{\mathcal{M}}(\mathcal{D}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}), \tag{1}$$

where $f(\mathcal{D}|\boldsymbol{\theta})$ is the likelihood of the data and c is a normalizing constant. Consider the following example: Imagine that the size of a given population is of interest, but unknown. A crude probability distribution of the population size is available from "mark and recapture" data. This distribution is the *a priori* information available and hence the prior for the analysis. Assume that a new effort is done and a collection of individuals from this population are sampled and genetic data becomes available, for instance the number of segregating sites at a given locus. In a Bayesian framework, the first step to do is to set up a model linking the population size (the parameter of interest) with genetic data, in this case the number of segregating sites. Then, the likelihood function is derived for this model. The best available estimate of the population size *a posteriori* of the genetic analysis is then, given by the Bayesian rule in equation 1, proportional to the product of the prior and the likelihood function, given the observed data $\mathcal{D}$ (the number of segregating sites in this example). In other words, the probability distribution of the parameters $\boldsymbol{\theta}$ before an experiment was conducted (our *a priori* knowledge) is challenged by the outcome of an experiment (data $\mathcal{D}$).

1.1.2 Approximate Bayesian Computations

Unfortunately, the evaluation of the likelihood function is far from trivial in complex models and in many cases even intractable. However, as initially proposed by Tavare et al. (1997), stochastic simulations can replace the likelihood: a candidate parameter vector $\boldsymbol{\theta}$ is sampled from the prior distribution and accepted if the simulated vector of summary statistics $\mathbf{s} = (s_1, \ldots, s_n)$ is sufficiently "close" to the observed summary statistics $\mathbf{s}_{obs}$ (if $\| \mathbf{s} - \mathbf{s}_{obs} \| < \boldsymbol{\epsilon}$, a fixed tolerance level). The distribution of the accepted parameter values is an estimate of the posterior. Therefore, if it is possible to simulate data (summary statistics) from a model, Bayesian inference is possible even if the likelihood is of unknown analytical form. But this at the cost of some approximation: the likelihood is assumed to be constant for all accepted summary statistic vectors. More recently, improvements on the rejection algorithm presented above have been proposed, including an MCMC without likelihood (Marjoram et al. 2003) and a Population Monte Carlo sampler PMC (Sisson et al. 2007). All these methods are generally subsumed under the term likelihood–free inference or Approximate Bayesian Computations ABC.

1.1.3 Summary Statistics

In order to evaluate the distance between the simulated data $\mathcal{D}'$ and the observed data $\mathcal{D}$, the data needs to be summarized in a quantitative form. This is easily done by calculating a set of summary statistics on the full data $\mathcal{D}$. The choice of summary statistics, however, is a tough one. Too few summary statistics may not capture the characteristics of the model, which leads to overall low power for parameter estimation. Too many summary statistics, on the other hand, may introduce random noise and eventually distort the estimation procedure (Wegmann and Excoffier 2008). We generally recommend to start with few, carfully picked summary statistics or to use linear combinations as proposed by Wegmann and Excoffier (2008). Some more detailed comments on this issue are given in Section 5.2 "Choosing Summary Statistics" on page 44.

1.1.4 Post–sampling Adjustment

The likelihood–free approach introduced above makes a tough assumption: the likelihood among the accepted summary statistic vectors has to be constant. While this assumption may hold for very small tolerance levels ϵ, it is likely to reduce the precision of the estimates in most situations. Luckily we can expect that the truncated model locally around $\mathbf{s}_{obs}$ will exhibit a simpler structure than the full model (for which likelihood calculations are impossible). Indeed, a linear relationship between parameters and summary statistics is often observed locally around $\mathbf{s}_{obs}$. Recently, Beaumont et al. (2002) introduced a post–sampling regression adjustment, taking the variation of the likelihood among the accepted summary statistic vectors into account, and greatly reduces the dependence on small tolerance values ϵ (and consequently speeding up the ABC estimation). `ABCtoolbox` uses an approach similar in spirit that naturally embeds into the standard Bayesian framework, which in turn allows the application of well-known Bayesian methodologies such as model selection via Bayes factors. The main assumption is that the likelihood follows a general linear model (GLM) among the accepted summary statistic vectors. Using standard regression techniques, the likelihood function can be estimated from the accepted parameter and summary statistic vectors and used to compute the posterior densities.

1.2 Understanding the General Workflow of `ABCtoolbox`

`ABCtoolbox` is a collection of programs that can be pipelined to estimate model parameters under various ABC algorithms. The package is designed to make use of grid computing and can be fully controlled via easy to understand input files or the command line. We recommend to use input files which can easily be modified and reused at any time. An ABC estimation is usually performed in two distinct steps: a large set of simulations is performed first and a subset close to the observed summary statistics $\mathbf{s}_{obs}$ is then used to estimate posterior distributions. The package incorporates two main programs for these steps (Figure 1): `ABCsampler` aims at producing a large collection of simulations, resulting in a matrix of model parameters and their associated summary statistics, and `ABCestimator` is then used to calculate the marginal posterior distributions from the stored simulations, with or without regression adjustment.

`ABCsampler` samples model parameter values from specified prior distributions, passes these values to an external simulation program, calls an external program calculating summary statistics on the simulated data, and finally writes the parameter values and the resulting summary statistics into a file. Interaction with external programs is flexible due to the possibility to pass values via the command line or via input files. This allows the use of simulation programs such as `SIMCOAL 2.0` (Laval and Excoffier 2004), `ms` (Hudson 2002) or `FREGENE` (Chadeau-Hyam et al. 2008). Of course, ABCsampler can also interact with a single program directly generating summary statistics from model parameters, such as `quantiNemo` (Neuenschwander et al. 2008). Details on how `ABCsampler` interacts with different simulation programs and how to configure `ABCsampler` to use a specific simulation program is outlined in Section 2.4 "Using different Simulation Programs" on page 13.

Importantly, `ABCsampler` is also fully flexible in the number of programs to be called per iteration (Figure 1), which allows the combination of various types of data. For instance, the same model parameters can be used to generate genetic and isotopic data using two different simulation programs, respectively. By calling a simulation program such as `SIMCOAL 2.0`

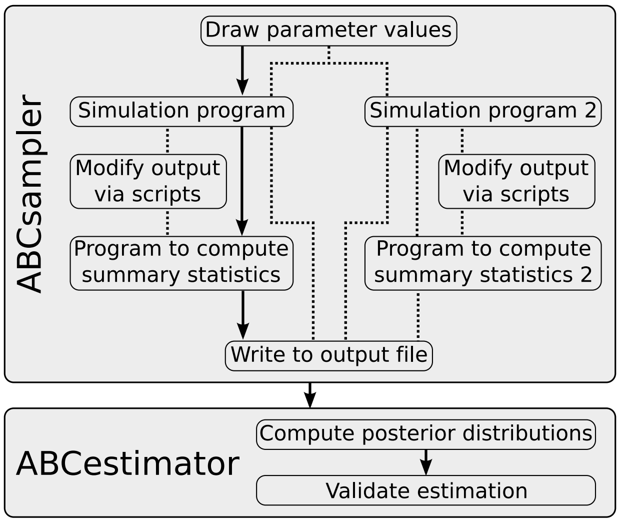

ABCsampler
Draw parameter values
Simulation program
Simulation program 2
Modify output via scripts
Modify output via scripts
Program to compute summary statistics
Program to compute summary statistics 2
Write to output file
ABCestimator
Compute posterior distributions
Validate estimation

Figure 1. ABC Flowchart *Flowchart describing the individual steps of an ABC estimation by* `ABCtoolbox`*. Black arrows indicate the standard approach. Alternative approaches including the use of two simulation programs are shown as dotted lines.*

several times, genetic markers with different mutational models (such as DNA sequences and microsatellites) or different ploidy levels can be used simultaneously. This feature is explained in detail in Section 2.7 "Several simulations per iteration" on page 18. Additionally, `ABCsampler` offers the possibility to modify the output of a simulation program prior to the summary statistics calculation via scripts, which allows, for instance, to reproduce observed patters of missing data (see Section 2.6 "Modifying files at every iteration with bash scripts or programs (optional)" on page 17).

Beside generating posterior distributions, `ABCestimator` is used for validation by analyzing a large set of pseudo-observed data sets at once. For each of these data sets, accuracy measures and the posterior quantiles of the known parameter values are computed, which are informative about potential estimation biases (Wegmann and Excoffier 2008).

1.3 Obtaining and Installing `ABCtoolbox`

`ABCtoolbox` can be obtained, together with this manual, from the software page of the Computational and Molecular Population Genetics Lab at the University of Bern: *www.cmpg.unibe.ch.*

1.3.1 Content of `ABCtoolbox`

Beside this manual and several example files, `ABCtoolbox` contains the following programs or scripts. For each piece of software, the section describing it's use is given in parentheses.

- `ABCsampler` (2)
- `ABCestimator` (3)
- `transformer` (4.1)
- `cumuldens` (4.2)
- `strStats` (4.3)
- `glm` (4.4)
- `plotPosteriors.r` (4.5);
- `findPLS.r` (5.2)

The package further ships with compiled versions of the programs `simcoal2` and `arlsumstat` for convenience. They do, however, not belong to the `ABCtoolbox` itself.

1.3.2 Licence

`ABCtoolbox` is free to use, distribute and modify under the terms of GNU General Public Licence as published by the Free Software Foundation; either version 3 of the licence, or any later version. In particular `ABCtoolbox` comes **without any warranty**. Note that both, `ABCsampler` and `ABCestimator` contain files from the `newmat11` library to perform matrix operations. The `newmat11` library is not published under the GNU General Public Licence but is published without any restriction to incorporate it into any software, either commercial or not. The `newmat11` library is available at *http://www.robertnz.net*. The programs `simcoal2` and `arlsumstat` do not belong to the package and are distributed on their own. Please check their respective websites at *www.cmpg.unibe.ch* for information on their licence.

1.3.3 Installing `ABCtoolbox`

Installing is straightforward: download the file `ABCtoolbox.gz` and unpack it by typing

```
$ tar -xzvf ABCtoolbox.tgz
```

on Linux or on Windows using any zipping software. The created folder will contain several subfolders: one containing the source code of all programs, one containing all precompiled binaries, one containing all scripts and an additional one with example files. Note that some Windows users need to install cygwin (*www.cygwin.com*) in order to use the precompiled binaries. In the folder with example files, however, we provide the standard cygwin `.dll` files such that the example may run on many Windows computers without installing cygwin. If the provided binaries do not work on your system simply enter the directories with the source code and compile the programs yourself. We successfully compiled all programs using `g++ 4.3` on Linux and Windows systems (using cygwin). Since file handling and program execution is different between different operating systems, not all functionalities of `ABCsampler` may be available in the Windows version if compiled with different compilers. This is namely the use of scripts as described in Section 2.6 "Modifying files at every iteration with bash scripts or programs (optional)" on page 17. To use the whole functionality use the compilation variable _GCC_ with the `g++` compiler.

1.4 Running `ABCtoolbox` – a quick start guide

`ABCtoolbox` is a series of computer programs for parameter estimation with various ABC algorithms including rejection sampling (Tavare et al. 1997), MCMC without likelihoods (Wegmann and Excoffier 2008) and PMC (Beaumont et al. 2009). The package also includes a program to perform a post–sampling adjustment, which allows for model selection via Bayes factors (Leuenberger and Wegmann 2009). The magic of performing ABC estimations is to combine these different pieces of software such as to build up the desired ABC algorithm. Here we introduce the basic setup for the standard rejection algorithm and the use of the standard post–sampling adjustment. We will use the example files provided in the subfolder `exampleFiles`. Please copy the executables of `ABCsampler` and `ABCestimator` into this subfolder when playing around with this example.

Step 1: The Model The first important step is to define the model in question. The model should capture the main features influencing the data, and yet be as simple as possible. It is very difficult to get unbiased estimates with models with more than about 15 parameters. Try to start with the most simple model possible. An extension is always possible. Once the model is set up, it has to be parameterized. Use natural parameters whenever possible. While ratios may be nice features, they are generally more difficult to estimate. As an example we will use here a simple population genetics model, given in Figure 2: Consider a population of size *N_NOW* that was larger in the past and changed its size *T_SHRINK* generations ago. The ancestral population size is given by *N_ANCESTRAL*. For this model all necessary example files are provided in the subfolder `exampleFiles`.

For each parameter of the model a prior distribution has to be defined. Prior distributions are probability distributions of the model parameters and should summarize the current (prior) knowledge of the parameters in question. Try to avoid too large prior ranges. If your prior covers several orders of magnitude, be aware that only very limited weight will be given to small values. In such cases, it is often advisable to use priors on the logarithmic scale.

The model parameters used and the corresponding prior distributions are specified in a specific file with the extension `.est`. The specific structure of this file is outlined in Section 2.3 "Defining prior distributions - `.est` File" on page 11. The corresponding `.est`–file is named `example.est`.

Step 2: External Programs: The program `ABCsampler` is used to generate a large set of

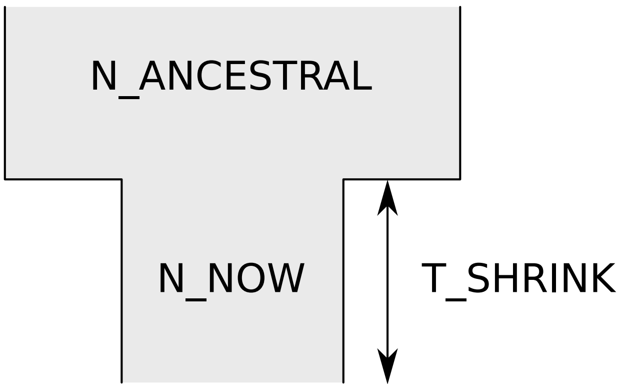


Figure 2. A simple population genetics model *This figure describes the model used as an example throughout the manual. Consider a population of size N_NOW that was larger in the past and changed its size T_SHRINK generations ago. The ancestral population size is given by N_ANCESTRAL. For this model all necessary example files are provided in the subfolder `exampleFiles`.*

simulations with parameter values drawn from the prior distribution. As mentioned above, `ABCsampler` interacts with one or several external programs. In most cases two programs are needed: one to simulate data, and one to compute the summary statistics. We will demonstrate the functionalities of `ABCsampler` using the two freely available programs `simcoal2` (Laval and Excoffier 2004) to generate data from our simple model, and the program `arlsumstat` to generate summary statistics on the output of `simcoal2`. While both programs ship with `ABCtoolbox`, they are distributed on their own with their own licence (see above). We nonetheless provide some guidelines on their usage in Section 5.4 "Using `ABCtoolbox` with `simcoal2` and `arlsumstat`" on page 50. The binaries necessary to play around with the provided example are already present in teh example folder.

Step 3: Summary Statistics / Observed Data: A set of summary statistics, on which the ABC estimation will be based, has to be defined. The potential summary statistics are given by the chosen program to compute summary statistics. The choice of summary statistics is nonetheless a tricky issue. As mentioned above, we generally recommend to start with few, carfully picked summary statistics or to use linear combinations as proposed by Wegmann and Excoffier (2008). Some more detailed comments on this issue are given in Section 5.2 "Choosing Summary Statistics" on page 44.

Once the set has been defined, compute these summary statistics on the observed data. Provide them finally in a specific file named `.obs`–file (see Section 2.9 "Passing the observed data: the `.obs` file" on page 21). Note that `ABCsampler` will only keep the summary statistics mentioned in the `.obs`–file, even if the program to compute the statistics outputs additional ones. The `.obs`–file for our example is named `example.obs` and is located in the subfolder `exampleFiles`.

Step 4: Configuring `ABCsampler`: `ABCsampler` is designed to make use of grid computing and can be fully controlled via an easy to understand `.input`–file or the command line. We recommend to use an `.input`–file, which can easily be modified and reused at any time. A detailed description on the format of the `.input`–file is given in Section 2.2 "`.input` File" on page 11. The `.input`–file of our example is named `example.input` and is located in the subfolder `exampleFiles`.

The most tricky part is to tell `ABCsampler` how to use the external programs. `ABCsampler` is very flexible in the way it interacts with external programs and almost any command line program can be used. Details on how to configure `ABCsampler` to use a specific program is outlined in Section 2.4 "Using different Simulation Programs" on page 13 and Section 2.5 "Using different Summary Statistics Programs" on page 16 for simulation programs and programs to compute summary statistics, respectively.

Note that the choice of the simulation program may have an influence on the parameterization of the model. For instance, simcoal2 requires the ancestral size to be specified relative size of the population. Fortunately ABCtoolbox offers the possibility to define some parameters as equations from other (see Section 2.3.3 "[COMPLEX PARAMETERS]" on page 12). We may therefore keep the original parameterization and simply add a new parameter which is the ratio of ancestral and current size.

ABCsampler also offers to use an MCMC without likelihood or a Population Monte Carlo algorithm. These algorithms require specific configurations, as is described in detail in Section 2.11 "Sampler Types" on page 22.

Step 5: Running sampler: Once ABCsampler has been configured successfully, it has to be launched in order to produce a large set of simulations:

```
$ ABCsampler example.input
```

This may take some time, depending on the number of simulations to perform. To speed up ABCsampler may be used on a grid (see Section 5.5 "Parallelizing ABCsampler" on page 51). The output of several runs is easily concatenated (see Section 5.5 "Parallelizing ABCsampler" on page 51). If ABCsampler detects a problem it will print an error message (see Section 2.13 "Possible error Messages" on page 27). It is therefore advisable to produce a few simulations on a host machine in order to check if everything is fine, before sending a job to the grid. Make sure all external programs used by ABCsampler are located in the running directory or their location is accurately listed in the .input–file. ABCsampler will wait for any external program to finish. This is problematic if an external program hangs (such as simcoal2 does, if its inputfile contains an error). Try to launch the external program directly to test if they are working fine.

Step 6: Computing Posterior Distributions: ABCestimator is used to compute posterior distributions, with or without post–sampling adjustment. If you don't have a specific reason we recommend to use the post–sampling adjustment, since it reduces the uncertainty of the estimates in general. ABCestimator is configured with a simple .input–file (see Section 3.2 ".input File" on page 32). An .input–file working for our simple example is named estimator.input and is provided in the subfolder exampleFiles. Simply launch ABCestimator as follows:

```
$ ABCestimator estimator.input
```

Make sure the file containing the performed simulations (from Step 5) is named example_output_sampling1.txt, which is the default name using the provided .input–file for ABCsampler.

Step 7: Visualization: ABCtoolbox includes an R script plotting posteriors directly from the output of ABCestimator. Simply launch this script on the command line as follows:

```
$ R --vanilla inputfile < plotPosteriors.r
```

where inputfile corresponds to the name of the ABCestimator .input–file used in Step 6. A detailed description on the use of this script is given in Section 4.5 "Plotting Posteriors" on page 43.

Step 8: Validation: Unfortunately, ABC always leads to posterior distributions, whether the underlaying assumptions are met or not. It is therefore important to perform some additional analysis to gain confidence in the results obtained. While all recommended validation steps are described in detail in Section 5.3 "Validation" on page 47, we show here how to perform two of them. If performing the post–sampling adjustment, ABCestimator computes a p–value describing the fit of the estimated general linear model to the observed data (see Section 3.8 "Testing Model Fit" on page 35). The p–value is reported in the output of ABCestimator on the same line as the marginal density. If this p–value is very low, it is likely that model used to generate the simulations is inadequate. The second validation step introduced here is to check for biased posteriors using pseudo–observed data sets (summary statistics generated under the model with known parameter values). If the parameter values for these pseudo–observed were randomly chosen from the prior distribution we expect the posterior quantiles (the position of the true values within the posterior distribution) to be uniformly distributed (see Section 5.3.2 "Checking for biased posteriors" on page 48). ABCestimator computes posterior quantiles if a file with true values is provided. The subfolder exampleFiles contains files with pseudoobserved data sets generated under our simple model with parameter values drawn from the prior distribution defined in `example.est`: `pseudoObserved.obs` and `pseudoObserved.true`. This folder also contains an `.input`–file for `ABCestimator` pre–configured to use these pseudo–observed data sets (`pseudoObs.input`). Simply launch `ABCestimator` as follows:

```
$ ABCestimator pseudoObs.input
```

`ABCestimator` will generate a file named `ABC_GLM_quantilesOfTrueParameters.txt` containing the posterior quantiles for each model parameter and for each pseudo–observed data set. The subfolder `exampleFiles` also contains a simple R script generating a pdf file with histograms and p–values based on a Kolmogorov–Smirnov test for each model parameter. Simply launch the script as follows (given that R is installed):

```
$ R -- --vanilla < analyzeQuantiles.r
```

See Section 5.3.2 "Interpreting the Distribution of Posterior Quantiles" on page 49 for explanations on how to interpret these histograms. But basically, everything is fine if all parameters pass the test (note that a Bonferroni–correction may be applied).

1.5 How to cite

We recommend users of `ABCtoolbox` to cite the publication of this package as well as the papers where the implemented methodology was published originally:
D. Wegmann et al. (2009). "ABCbox1.0: A Toolkit to perform various ABC Algorithms". In: *Bioinformatics*

When using an MCMC without likelihoods, please cite additionally:
P. Marjoram et al. (2003). "Markov chain Monte Carlo without likelihoods". In: *Proceedings Of The National Academy Of Sciences Of The United States Of America* 100.26. Pp. 15324–15328
D. Wegmann and L. Excoffier (2008). "Efficient Approximate Bayesian Computation coupled with Markov Chain Monte Carlo without likelihood". In: *Genetics*

When using the PMC sampler, please cite additionally:
S. A. Sisson et al. (2007). "Sequential Monte Carlo without likelihoods". In: *Proceedings of the National Academy of Sciences of the United States of America* 104.6. Pp. 1760–1765
M Beaumont et al. (2009). "Adaptivity for approximate Bayesian computation algorithms: a population Monte Carlo approach". In: *Biometrika*

When using any ABC–GLM regression adjustment:
Christoph Leuenberger and Daniel Wegmann (2009). "Bayesian Computation and Model Selection in Population Genetics". In: *genetics*

2 Using ABCsampler

As mentioned above, the program `ABCsampler` is designed to interact with two other programs, one to perform simulations and one to calculate summary statistics. The principle cycle, which is repeated many times, proceeds as follows:

1) Parameter values for the model in question are drawn from the appropriate prior distributions.
2) These parameter values are then passed to the simulation program resulting in a simulated data set.
3) Then, the program calculating the summary statistics is launched to calculate the desired statistics on the simulated data set.
4) Finally, the resulting summary statistics are stored in a convenient format.

Note that this cycle may be within an MCMC framework or not, specified by the parameter `samplerType` (see Section 2.11 "Sampler Types" on page 22).

2.1 Launching ABCsampler

`ABCsampler` is a command line program launched with one mandatory argument, the name of the `.input` file. All necessary parameters are specified in the `.input` file (see Section 2.2 "`.input` File" on page 11). Suppose the inputfile is named `name.input`, the program has to be launched as follows:

```
$ ABCsampler name.input
```

Additionally, parameters may be passed on the command line. In that case, parameters given in the `.input` file will be overwritten. The required syntax is as follows:

```
$ ABCsampler name.input parameter=value
```

where `parameter` is the name of a parameter and `value` is the corresponding value. For instance,

```
$ ABCsampler name.input nbSims=10000
```

launches `ABCsampler` to produce 10,000 simulations.

There is no restriction for the number of parameters to be passed that way. Note, however, that the program always requires the name of an existing `.input` file as the first parameter. `ABCsampler` writes a `.log` file specifying all the parameters used, indicating the progress of the simulations and printing important messages in case of an error (see Section 2.13 "Possible error Messages" on page 27).

2.2 .input File

An `.input` file is the most convenient way to specify parameters for `ABCsampler`. One reason is that the parameters used are stored in a reusable fashion. Basically, an `.input` file is a simple collection of pairs of parameter tags and corresponding values, enriched with comments. While each parameter–value pair has to be at the beginning of a new line, the order is of no importance. Comments may either be added after a parameter–value pair or given on a line on their own. Note that all comments must be proceeded with a double slash "//". Empty lines may be present anywhere within the file. A minimal `.input` file is given in Figure 3. A complete list of all available parameter tags is given in Section 2.12 "All Available Parameters for `ABCsampler`" on page 25.

2.3 Defining prior distributions - .est File

The `.est` file is the key file where model parameters and corresponding priors are defined. It consists of three sections, each of them is started by one of the following identification tags: `[PARAMETERS]`, `[RULES]` and `[Complex PARAMETERS]`. An example of a simple `.est` file is given in Figure 4. The name of the `.est` file is a mandatory parameter for `ABCsampler` and is passed with the `estName` tag in the `.input` file or via the command line (see above).

2.3.1 [PARAMETERS]

The section `[PARAMETERS]` is the only mandatory section in an `.est` file. It defines the parameters of the model and the corresponding priors. Each line starts by indicating whether a parameter has to be treated as an integer (1) or not (0). Then the parameter name follows. It may contain any character except spaces, tabs and a few characters with special meaning (see Section 2.4.1 "Hyper Priors" on page 14). The parameter's name is followed by the definition of the prior, starting by the distribution tag and two to

```
//a simple inputfile for ABCsampler
samplerType standard
//only uniform priors
estName test_uniform.est
obsName test.obs
outName test_output
nbSims 100
writeHeader 1
//settings regarding the simulation program
simulationProgram simcoal2
simInputName test.par
simParam FILENAME#1#1#0 //see simcoal2 manual for details
//program to calculate summary statistics
sumStatProgram arlsumstat
simParam SIMINPUTNAME_0.arp#SSFILENAME#0#1 //see arlsumstat manual for details
```

Figure 3. An input file *This is an example of a very simple `.input` file for the program `ABCsampler`. Each line represents a pair of a parameter and a corresponding value. A complete list of all possible parameters is given in Section 2.12 "All Available Parameters for `ABCsampler`" on page 25. Note that comments, indicated with //, are possible on whole lines or after the declaration of a parameter.*

four specific parameters. Currently, four different distributions are available:

unif An uniform distribution $X \sim [a, b]$. This distribution is invoked with `unif a b`.

logunif A log-uniform distribution, corresponding to the uniform distribution $\log(X) \sim [\log(a), \log(b)]$. This distribution is invoked with `logunif a b`

norm A truncated normal distribution with $X \sim \mathcal{N}(\mu, \sigma)$ truncated such that $a \leq X \leq b$. Such a distribution is specified with `norm` μ σ `a b`.

lognorm A log-normal distribution such that $\log(X) \sim \mathcal{N}(\hat{\mu}, \hat{\sigma})$ with $\hat{\mu}$ and $\hat{\sigma}$ chosen such that the resulting distribution has mean μ and standard deviation σ in the natural scale. The distribution is truncated at $a \leq X \leq b$. Such a prior distribution is specified with `lognorm` μ σ `a b`.

2.3.2 [RULES]

In some cases it may be desirable to limit the prior distribution of a parameter by the value of another parameter. For instance, the simple model implied in the example `.est` file in Figure 4 corresponds to a model where a population suffered from a reduction in size (`N_NOW < N_ANCESTRAL`). In such cases it is possible to specify a rule in the section `[RULES]` of the `.est` file. Possible signs for rules are $>$ or $<$. On the left side of the relation sign, a name of a model parameter specified in the section "`[PARAMETERS]`" is required. This model parameter will be updated if the rule is not matched [1]. On the right hand side of the relation sign either a model parameter or an equation including numbers and model parameters are allowed. A detailed description on equations will be given in Section 2.3.3 "`[COMPLEX PARAMETERS]`" on page 12. Note, however, that only model parameters from the section `[PARAMETERS]` may be used. Be aware that `ABCsampler` will only accept parameter vectors satisfying all rules. If a rule is violated, the whole parameter vector will be discarded and a new one generated form the prior distribution.

2.3.3 [COMPLEX PARAMETERS]

The last section, specified with the tag `[COMPLEX PARAMETERS]`, allows the declaration of model parameters, which are calculated from other model parameters. In our example, for instance, the ancestral population size `N_ANCESTRAL` has to be passed relative to the current population size `N_NOW` to `simcoal2`. Since it was preferred in this

[1]Basically, new values are drawn from the prior distribution of this parameter, until the rule is matched.

```
//Priors for the test model
[PARAMETERS]
1 N_NOW unif 100 10000
1 N_ANCESTRAL unif 100 10000
1 T_SHRINK unif 10 100
0 MUTATION norm 0.0001 0.0010 0.0005 0.0002

[RULES]
N_NOW < N_ANCESTRAL

[COMPLEX PARAMETERS]
0 N_ANCESTRAL_RELATIVE = N_ANCESTRAL / N_NOW
```

Figure 4. An `.est`**– file** *This is an example of a very simple* `.est`*– file for the program* `ABCsampler`*. Among the three sections, all embraced with squared brackets, only "[PARAMETERS]" is mandatory. For a detailed description see Section 2.3 "Defining prior distributions -* `.est` *File" on page 11.*

case to define the prior distribution in absolute values, we need to define a new model parameter as an equation of others. Each line in this section defines a new model parameter and starts by indicating whether the parameter has to be treated as an integer (1) or not (0), followed by the desired name of the model parameter. After that, a "=" sign is followed by an equation.

Equations may contain numbers and model parameters specified earlier in the `.est` file, that is, in the section `[RULES]` or in the same section, but above. Brackets may be used as well as the four standard operators `+`, `-`, `*` and `/`. Additionally, four functions are currently defined:

log(x**)** calculates the natural logarithm of x.
log10(x**)** calculates the logarithm of x to the base of 10.
exp(x**)** raises e to the power of x.
pow10(x**)** raises 10 to the power of x.
abs(x**)** returns the absolute value of x.

Each function has to be associated with brackets. An example of a more complex equation, based on the model parameters defined in Figure 4, is the following: `LOG_THETA=log(4*N_NOW*MUTATION)`. Since all equations are interpreted from left to right, be careful to use necessary brackets. For instance, `1+T_SHRINK/N_NOW` $\neq$ `1+(T_SHRINK/N_NOW)`. Note that any string which does not match a model parameter's name will be converted into a floating point number. Therefore, misspelled model parameters will be treated as 0.0. Also, `ABCsampler` does not report an error if an equation contains a model parameter not specified earlier. But `ABCsampler` will always use the value of the previous iteration to calculate the equation. Note that the function `abs` can be used to produce an if–like structure since

$$\frac{1}{2}\left(1+\frac{A-B}{abs(A-B)}\right) \qquad (2)$$

is 1 if $A > B$ and 0 otherwise.

2.4 Using different Simulation Programs

The simulation program to use is a mandatory parameter for `ABCsampler` and is passed with the `simulationProgram` tag in the `.input` file or via the command line (see Section 2.1 "Launching `ABCsampler`" on page 11). In principle, any simulation program can be coupled with `ABCsampler`, given that the program can receive model parameters from `ABCsampler`. This is crucial since `ABCsampler` draws the parameters of the model from the prior distributions and passes them to the simulation program. There are two ways how `ABCsampler` can pass model parameter values to a simulation program: 1) via files or 2) via the command line. Many simulation programs require an input file where parameters are specified. For instance, `simcoal2` requires a `.par` file where the parameters and the model to simulate are specified (see online documentation for `simcoal2` at *www.cmpg.unibe.ch*). If the name of an input file for the simulation program is given (with the tag `simInputName`), then `ABCsampler` parses this file and replaces all occurrences of prior or complex prior names with its current value. In the `.par` file shown in Figure 5, in the line "`T_SHRINK 0 0 1 N_ANCESTRAL_RELATIVE 0 0`", for instance, the two tags `T_SHRINK` and `N_ANCESTRAL_RELATIVE` are replaced by the current value of these two model parameters. Remember that all model parameters and their associated prior distributions are defined in the `.est` file (see Section 2.3 "Defining prior distributions - `.est` File" on page 11). Some simulation programs require several files to be parsed. In order to have `ABCsampler` parsing several files, simply specify several file names with the parameter `simInputName`, delimited with a "#".

The arguments to pass to the simulation program are specified with the parameter `simParam`, either in the `.input` file or on the command line. Different arguments are separated by "#". The individual arguments may be any character string, but may not contain a "#" nor a ";". There are two tags with special meaning:

SIMINPUTNAME Instead of this tag, the name of the input file defined with `simInputName` will be passed. Since the `.input` file for the simulation program will be parsed, the name passed is the name of the parsed file, which is similar to the one specified with `simInputName`, but with a `-temp` added before the extension. If several files are specified, use the tags SIMINPUTNAME0, SIMINPUTNAM1 etc. Actually, SIMINPUTNAME is a synonym for SIMINPUTNAME0.

SIMNUM Instead of this tag, the number of the current simulation is passed.

As mentioned above, model parameters may also be passed via the command line. This is done simply by putting the name of the model parameter as an argument. For instance, `simParam SIMINPUTNAME#N_NOW` will tell `ABCsampler` to launch the simulation program with two arguments: firstly, the name of the input file for the simulation program and secondly, the current value of the prior `N_NOW`. If a model parameter is constant for all simulations, its value may be passed via a file or via the command line as a character. For instance, `simcoal2` requires beside the name of a `.par` file three additional arguments: the number of simulations to perform, whether genotypic or haplotypic data is generated and whether the gametic phase is known. Of course, `simcoal2` is requested to perform a single simulation per iteration. Also, whether haplotypic or genotypic data is produced does in most cases not change between iterations. The gametic phase is most likely either known or unknown for all simulations aswell. For `simcoal2` the parameter `simParam` may therefore be specified like this: `simParam SIMINPUTNAME#1#1#0`.

2.4.1 Hyper Priors

`ABCsampler` also allows for different types of hyper priors to be used. In such cases, the current value of one model parameter specified in the `.est` file is used to shape the prior distribution from which individual values are then drawn. For instance, a prior on the average mutation rate could be defined as a model parameter in the `.est` file, but individual mutation rates for different loci might follow a normal distribution. The distribution of individual mutation rates corresponds to the prior distribution, the distribution of the mean of the mutation rate thus to the hyper prior. In order to use a model parameter from the `.est` file as a hyper prior, the tag of this model parameter is preceded by an additional parameter enclosed by a specific character identifying the type of prior to be used for individual values. An example: while parsing the input file for the simulation program, the phrase `$0.0001$MUTATION` will be replaced by `ABCsampler` by a random value from a normal distribution with mean equal to the current value of the model parameter `MUTATION` and standard deviation 0.0001 (see Figure 5). Therefore, the prior distribution on the model parameter `MUTATION` is in fact a hyper prior. Note that the parameter between the special character may also be a tag of a model parameter specified in the `.est` file. As an example, one might specify an additional hyper prior on the standard deviation of the normal distribution for individual mutation rates by defining a model parameter in the `.est` file and using its tag as parameter for the normal distribution (between the two "$"). When using model parameters as hyper priors, the following prior distributions are currently available:

Gamma (%) The phrase `%ARG%PARAMETER` will be replaced by a value drawn from a

```
//Parameters for simcoal2
1 samples to simulate
//Deme sizes
N_NOW
//Sample sizes
20
//Growth rates
0
//Number of migration matrix
0
//Historical events
1
T_SHRINK 0 0 1 N_ANCESTRAL_RELATIVE 0 0
//Number of independent chromosome
3 1
//Number of contiguous linkage blocks
1
//Data type, No.  loci, Recomb.  rate, optional parameters
MICROSAT 1 0.000 $0.0001$MUTATION 0 0
//Number of contiguous linkage blocks
1
//Data type, No.  loci, Recomb.  rate, optional parameters
MICROSAT 1 0.000 $0.0001$MUTATION 0 0
//Number of contiguous linkage blocks
1
//Data type, No.  loci, Recomb.  rate, optional parameters
MICROSAT 1 0.000 $0.0001$MUTATION 0 0
```

Figure 5. A par file *This is an example of a `.par` file for the program `simcoal2` specifying the simple population genetics model described in the text. `ABCsampler` parses this file prior to launching `simcoal2`While parsing, `ABCsampler` replaces all occurrences of prior tags specified in the `.est` file by the current value of this parameter. Also, hyperpriors may be used as is done here for the individual mutation rate of the different loci. See text for more explanations.*

gamma distribution $\Gamma(ARG, \frac{ARG}{CUR})$, where CUR is the current value of the parameter `PARAMETER` specified in the `.est` file. If `ARG` is the name of another model parameter, its current value will be used. Otherwise, it will be converted into double. Note that this leads to zero if `ARG` is a character string different from any model parameter specified in the `.est` file. Such a setting was introduced with $ARG = 20$ in Excoffier et al. (2005) for individual mutation rates.

Beta (&) The phrase `&PARAMETER` will be replaced by a value drawn from a beta distribution $\beta(a, b)$, where $a = 0.5 + 199 \times CUR$ and $b = \frac{a\times(1-CUR)}{CUR}$ and CUR is the current value of the parameter `PARAMETER` specified in the `.est` file. Note that there is no additional parameter to pass for the Beta distribution. This type was used in Excoffier et al. (2005).

Normal ($) The phrase `$ARG$PARAMETER` will be replaced by a value drawn from a normal distribution $\mathcal{N}$(`CUR`, `ARG`) with mean `CUR`, which is the current value of the parameter `PARAMETER` specified in the `.est` file, and standard deviation `ARG`. If `ARG` is the name of another model parameter, its current value will be used as the standard deviation. Otherwise, it will be converted into double. Note that this leads to zero if `ARG` is a character string different from any model parameter specified in the `.est` file.

Truncated Normal (!) Different to the normal hyperprior mentioned above, this distribution is truncated at zero and therefore only returning positive values ($\geq$ 1 if integers are requested). To be specific, the phrase `!ARG!PARAMETER` will be replaced by a value drawn from a normal distribution $\mathcal{N}$(`CUR`, `ARG`) with mean `CUR`, which is the current value of the parameter `PARAMETER` specified in the `.est` file, and standard deviation `ARG`. If `ARG` is the name of another model parameter, its current value will be used as the standard deviation. Otherwise, it will be converted into double. Note that this leads to zero if `ARG` is a character string different from any model parameter specified in the `.est` file.

Lognormal (#) The phrase `#ARG#PARAMETER` will be replaced by the value 10^x, where x is drawn from a normal distribution $\mathcal{N}$(`CUR`, `ARG`) with mean CUR, which is the current value of the parameter `PARAMETER` specified in the `.est` file and standard deviation ARG. Note that both, CUR and ARG are in log scale. If `ARG` is the name of another model parameter, its current value will be used as the standard deviation. Otherwise, it will be converted into double. Note that this leads to zero if `ARG` is a character string different from any model parameter specified in the `.est` file.

Note that hyper priors are currently not supported when prior values are passed to the simulation program on the command line.

2.4.2 INTERNALGLM

`ABCtoolbox` includes a small simulation program, termed `glm`, designed to play around with the different sampler types (see Section 4.4 "Using `glm`" on page 42). Actually, `ABCsampler` features an internal version of the same program, which can speed up the usage of this tool massively. INTERNALGLM behaves like a true simulation program and has also to be specified in `ABCsampler` as such. In order to use it, simply set the argument of the parameter `simulationProgram` to "INTERNALGLM" and specify the arguments to pass to it with `simParam`, just as for any other simulation programs. INTERNALGLM takes exactly the same arguments as the program `glm` (see Section 4.4 "Using `glm`" on page 42).

2.5 Using different Summary Statistics Programs

The simulation program used must either write a file with summary statistics directly (see Section 2.5.2 "Running `ABCsampler` without a program to calculate summary statistics" on page 17) or a file which can be analyzed by another program to calculate summary statistics. In our simple example we use `simcoal2` to simulate genetic data under a given set of parameters. The output of `simcoal2` serves directly as an input file for `arlsumstat`, which calculates then the summary statistics from the simulated data. The program to calculate summary statistics is indicated with the parameter `sumStatProgram`. Note that this is an optional parameter. If not defined, `ABCsampler` will assume that there is no need to launch a specific program to calculate summary statistics. This may either be the case if the simulation program writes a file with the summary statistics itself or if a bash script is called to do so (see Section 2.6 "Modifying files at every iteration with bash scripts or programs (optional)" on page 17).

If a program to calculate summary statistics is defined, most likely some arguments have to be passed to this program. These parameters are specified with the parameter `sumStatParam`, which has a similar syntax as the tag `simParam` defined above. Different arguments are separated by "#" and the individual arguments may be any character string but may not contain a "#" nor a ";". There exist three tags with special meaning:

SIMDATANAME Instead of this tag, the name of the file with the simulated data will be passed. This name is defined with `simDataName` (see below).

SSFILENAME Instead of this tag, the name of the file to which the program is requested to write the summary statistics is passed. This name is defined with `sumStatFile` (see below).

SIMNUM Instead of this tag, the number of the current simulation is passed.

The name of the file with the simulated data written by the simulation program has to be specified with the parameter `simDataName`. In many cases the name is a derivate of the input file name for the simulation program, that is, if only prefixes and suffixes are added. In such cases it is advisable to use the tag `SIMINPUTNAME` as an identifier for the part of the filename identical to the name of the input file of the simulation program (without extension). For instance, the name of the output file written by `simcoal2` is identical to the input file, except the suffix "_0" and the new extension `.arp`. This may be coded like this: `simDataName SIMINPUTNAME_0.arp`. Of course, the file name of the simulated data may also be given in full. If several simulations are performed per iteration, however, the tag `SIMINPUTNAME` becomes especially handy (see Section 2.7 "Several simulations per iteration" on page 18).

2.5.1 File with computed summary statistics

The name of the resulting file, from which `ABCsampler` will read the calculated summary statistics, can be specified with the parameter `sumStatFile`. If the parameter `sumStatFile` is not provided, the default value `summary_stats_temp.txt` is assumed. This file name may be passed to the program calculating the summary statistics with the tag "`SSFILENAME`" (see above). Note that `ABCsampler` requires that the file of summary statistics consists of two lines: a first line with the names of the summary statistics and a second line with the corresponding values. The different columns have to be separated either by a tab or a whitespace. If the program computing summary statistics does not provide a file with these specification, a script may be launched at the end of every iteration, bringing the summary statistics in the right form (see Section 2.6 "Modifying files at every iteration with bash scripts or programs (optional)" on page 17).

2.5.2 Running `ABCsampler` without a program to calculate summary statistics

If no program to calculate summary statistics is specified with the parameter `sumStatProgram`, `ABCsampler` assumes that the simulation program (or a script / program launched after the simulation were performed, see below) provides a file with the computed summary statistics. If the computed summary statistics are not available in the requested form, `ABCsampler` is not able to read them correctly (see above). In order to deal with this, a script may be launched at the end of every iteration, bringing the summary statistics in the right form.

2.6 Modifying files at every iteration with bash scripts or programs (optional)

`ABCsampler` allows for bash scripts or programs to be launched before and after each simulation and before and after the computation of summary statistics. This provides a possibility to:

1) change the input file of the simulation program before it is parsed by `ABCsampler`,
2) modify the output of the simulation program before passing it to the program computing summary statistics, and
3) modify the output of the program computing summary statistics before it is read by `ABCsampler`.

Note that this feature is only available on a Unix platform and if `ABCsampler` was compiled with the "_GCC_" compiler variable (see Section 2.14 "Compilation" on page 31). The scripts or programs to be launched are given with the parameters `launchBeforeSim`, `launchAfterSim`, `launchBeforeSS` and `launchAfterSS`. The arguments for the scripts / programs are defined with `launchBeforeSimParam`, `launchAfterSimParam`, `launchBeforeSSParam` and `launchAfterSSParam`. Individual arguments are separated by "#" and the arguments may be any character string but may not contain a "#" nor a ";". Just as for the simulation program, for the scripts indicated with `launchBeforeSim` and `launchAfterSim`, the same tags with special meaning exist and prior values may be passed by putting their name as arguments (see Section 2.4 "Using different Simulation Programs" on page 13). For the scripts indicated with `launchBeforeSSParam` and `launchAfterSSParam`, the same tags with special meaning exist as for the program calculating the summary statistics (see Section 2.5 "Using different Summary Statistics Programs" on page 16).

The script or program indicated with `launchBeforeSim` will be launched after the update of parameter values from the priors but before the input file for the simulation program is parsed. This allows to change the input file of the simulation program, according to priors. Suppose we aim at simulating an island model with a prior on the number of islands with simcoal2. The number of islands can not be passed simply to simcoal2, since it requires each population to be specified explicitly. We could, however, write a small script generating input files for simcoal2, tell ABCsampler to launch this script in each iteration and to pass the number of islands drawn from a prior distribution to this script as an argument.

The script or program specified with launchAfterSim will be executed just after the simulation program. This allows to change the output of the simulation program before calculating summary statistics. Suppose there are some parts of the data missing in the observed data set. Unfortunately simcoal2 does not allow to simulate missing data. A workaround may be to write a script modifying the output of simcoal2 and to set some parts of the data as missing data before calculating the summary statistics.

The script or program specified with launchBeforeSS will be executed just before the program to calculate summary statistics is launched. The scripts specified with launchAfterSim and launchBeforeSS differ mainly in the possible arguments (tags with special meaning). A striking difference between these scripts exists only if several simulations are performed per iteration (see below).

The script or program specified with launchAfterSS will be executed after the termination of the program calculating the summary statistics, but before ABCsampler attempts to read the computed summary statistics. This is especially handy if the output of the program calculating summary statistics does not match the required specifications (see Section 2.5.1 “File with computed summary statistics” on page 17).

Before launching the loop of simulations, ABCsampler performs a single simulation for testing purposes. That way ABCsampler tests for the existence of all necessary files and adjusts internally the data storage system. This behavior is of importance if scripts are used to extract some data, since the script will be called once more than there are simulations outputted by ABCsampler. Suppose you use a script to extract some parts of the simulated data every iteration and store this data as an additional line in a file. This file will consist of one simulation more than is specified with the parameter nbSims, since the script will also be launched on the first simulation used for testing and adjusting. To get rid of the first line of a text file is easy under a Unix system, just use

```
$ tail -n+2 filename > new_filename
```

2.7 Several simulations per iteration

It is possible to tell ABCsampler to launch several simulation per iteration. The parameter runsPerParameterVector controls the number of repeats with exactly the same parameter vector. If runsPerParameterVector > 1, several simulations with the same model parameter values will be launched per iteration. The simulated summary statistics of all these simulations are outputted. In many cases, however, it may be desirable to launch several simulations per iteration with different simulation programs or different ways to compute summary statistics. This may be required, for instance, if the simulation program or the program computing summary statistics are not able to deal with the whole data set. As a population genetics example, suppose the observed data consists of DNA sequences and micro satellites (STRs). While simcoal2 can simulate both marker types at once, arlsumstat is not capable of analyzing them in one run. Therefore we request ABCsampler to run two independent simulations, one with DNA sequences and one with STRs. For a detailed description on how to achieve this, see Section 5.7 “Using different Marker Types with simcoal2” on page 52.

ABCsampler may be ask to 1) call the simulation program several times or to 2) call the program computing summary statistics several times independently (see below for a detailed description on how to chieve this). In each iteration ABCsampler performs the following steps:

1) New values for all model parameters are drawn from their respective prior distributions.
2) The following steps are repeated for all files given by the parameter simInputName:
    2.1) If requested, the script or program defined with launchBeforeSim is launched.
    2.2) The input file for the simulation program is parsed.
3) Then, the following steps are repeated for all files given by the parameter simInputName:
    3.1) The simulation program, specified with

simulationProgram, is launched.
   3.2) If requested, the script or program defined with launchAfterSim is launched.
4) Then, the following steps are repeated for all files given by the parameter simDataName:
   4.1) If requested, the script or program defined with launchBeforeSS is launched.
   4.2) If requested, the program to compute summary statistics is called.
   4.3) If requested, the script or program defined with launchAfterSS is launched.
   4.4) ABCsampler reads the computed summary statistics

Note the difference between the scripts indicated with launchAfterSim and launchBeforeSS: while the first is launched for each simulation step, the second is launch for each step where summary statistics are computed. Also be aware that the two modes of generating several simulations per iteration, namely using the parameter runsPerParameterVector or one of the modes described below, are by no means exclusive. ABCsampler repeats the whole loop depicted above runsPerParameterVector times.

2.7.1 Several calls of the simulation program per iteration

It is possible to tell ABCsampler to launch several simulations per iteration. This is specified by giving several filenames with the parameter simInputName, delimited by a ";". For instance, simInputName name1.par;name2.par will tell ABCsampler to launch the simulation program twice, once for each file specified.

Be aware that in every iteration, all data will be simulated first, before any attempt to calculate summary statistics. Therefore, ensure that the program generating simulated data sets does not override previous results in the iteration. This can be achieved in two different ways:

- If possible, the name of the resulting data file may be passed to the program performing the simulation (see Section 2.4 "Using different Simulation Programs" on page 13).
- A script may be called after each simulation to change the name of the file with the generated data (see Section 2.6 "Modifying files at every iteration with bash scripts or programs (optional)" on page 17).

It is not necessary to use the same simulation program for the different files, simply specify several programs with the parameter simulationProgram by delimiting them with a ";". If only one simulation program is given, the same one will be used for all files. If you specify several simulation programs, however, their number has to match the number of files specified with simInputName. Also, you have to specify the arguments to pass with the parameter simParam for each simulation program, even if the arguments are the same for the different programs.

Similar, also the arguments passed to the simulation program(s) may be specified differently for the different files, again by giving several argument lists by the parameter simParam, separated by a ";". Again, if only one argument list is given, the same arguments will be passed to all simulations. Note that in this case, the predefined tags may be of special significance (see Section 2.5 "Using different Summary Statistics Programs" on page 16).

Finally, if no input file for the simulation program is used, several simulations per iteration are still feasible by specifying several simulation programs with the parameter simulationProgram or several argument lists with simParam, or both. Note that the same simulation program or argument list may also be given several times.

While it is possible to specify a single script to be launched before the simulation program (defined by launchBeforeSim), several argument lists may be given for this script with the parameter launchBeforeSimParam. Again individual argument lists are delimited by a ";". The same is true for the script specified with launchAfterSim and the corresponding argument lists given by launchAfterSimParam.

2.7.2 Several summary statistics computations per iteration

It is also possible to call the program computing summary statistics several times per iteration. In fact, this is in most cases intended, if several simulations are performed per iteration (see above). However, it may also be beneficial, if only one simulation is performed. In a population genetics setting, ABC AMERICAS PAPER, for instance, computed summary statistics for different groups of populations. They used a script after each simulation (see Section 2.6 "Modifying files at every iteration with bash scripts or programs (optional)" on page 17) to produce several files with the same simulated data, but differing in the ways populations were grouped. Summary statistics were then computed on each of these files.

In order to tell `ABCsampler` to launch the program computing summary statistics on several simulate data files, just provide the names of all these files with the parameter `simDataName`, delimited by a ";". Also, different programs computing summary statistics may be used for different simulated data files. This can be specified by providing the name of the program for each simulated data file by the parameter `sumStatProgram`, again delimited by a ";". Different arguments for these programs can be given by the parameter `sumStatParam` in a similar way (see also Section 2.7.1 "Several calls of the simulation program per iteration" on page 19). If only one program is given, it will be used for all simulated data files. Different arguments may still be specified for each file containing simulated data. Of course, if several programs or argument lists are specified, their number has to match the number of files with simulated data sets. Note that `ABCsampler` requires a file with the observed data for each simulated data file. These files are specified with the parameter `obsName` and delimited by a ";" (see Section 2.9 "Passing the observed data: the `.obs` file" on page 21).

A script or program defined with the parameter `launchAfterSS` will be called immediately after the program computing summary statistics finished. It will also be called for each simulated data file, if `sumStatProgram` is not defined.

As shown in the outline of an iteration (see Section 2.7 "Several simulations per iteration" on page 18), `ABCsampler` attempts to read a file with the computed summary statistics for each file given by `simDataName` independently. This requires the program calculating summary statistics (or the program / script given by `launchAfterSS`) to replace the file specified with the parameter `sumStatFile` everytime it is called. It is currently not possible to specify several names with `sumStatFile`. The parameter `separateOutputFiles` controls, if all calculated statistics from different files given by `simDataName` are written into a single file (`separateOutputFiles 0`) or not (`separateOutputFiles 1`). If separate output files are requested, each will contain the columns with the parameters used.

While it is possible to specify a single script to be launched before the program calculating summary statistics (defined by `launchBeforeSS`), several argument lists may be given for this script with the parameter `launchBeforeSSParam`. Again, individual argument lists are delimited by a ";". The same is true for the script specified with `launchAfterSS` and the corresponding argument lists given by `launchAfterSSParam`.

2.8 Linear transformed statistics

The choice of summary statistics is a difficult one. Wegmann and Excoffier (2008) proposed to transform the computed summary statistics into PLS components. While a detailed discussion on different ways to choose summary statistics is given in Section 5.2 "Choosing Summary Statistics" on page 44, we outline here the possibilities to use linear transformations of statistics during an MCMC chain. Note that this is currently not possible if several output files are written as controlled by `separateOutputFiles` (see Section 2.7.2 "Several summary statistics computations per iteration" on page 19).

The linear combinations of statistics to use are defined in an additional file, the name of which is specified by the parameter `linearCombName`. This file contains, besides the mean and standard deviation of the statistics, the transformation matrix. Each line specifies the values for one statistics, starting by the name of the statistics, the mean and the standard deviation followed by the contribution of the statistics to the different linear combinations. In each iteration, before calculating the distance, `ABCsampler` standardizes each statistics using the provided moments, and the linear combinations are computed by summing the values of each statistics multiplied by its provided weight. The parameter `stdLinearCombForDist` controls whether (1) or not (0) the resulting linear combinations should be standardized before calculating the distance. See Section 5.2 "Choosing Summary Statistics" on page 44 for a discussion on this subject. `stdLinearCombForDist` is an optional parameter with default value 0.

```
//Name mean  SD     PCA1    PCA2    PCA3
STAT_A 0.456 1.234  0.4556 -0.1272  0.5542
STAT_B 5.825 0.199  0.5745  0.1397  0.9147
STAT_C 0.745 0.524 -0.4656  0.1332  0.0125
STAT_D 6.112 4.433 -0.8224 -0.7172 -0.4552
```

Figure 6. A file specifying a linear transformation *This is an example of a file specifying linear combinations (here PCA's) of statistics that are used to calculate the distance between observed and simulated data sets. Lines starting with a double slash are ignored.*

The mean and standard deviation is not calculated from the calibrating simulations because the statistics should be standardized the same way they were when defining the linear combinations. Note that this also offers the possibility not to standardize that statistics at all. Simply set the mean to 0 and the standard deviation to 1. Neither The statistics provided in the `.obs` file nor in the calibration file should be transformed, this will be done by `ABCsampler` automatically. A simple file with linear combinations is given in Figure 6. Note that lines starting with a double slash are ignored by `ABCsampler`.

It may be desirable in some cases to linearize the relationship between model parameters and statistics before transforming them linearly. `ABCsampler` offers the possibility of applying a Box–Cox transformation to the statistics:

$$s_{BoxCox} = \frac{s^\lambda - 1}{\lambda(\mathrm{GM}(s))^{\lambda-1}}, \tag{3}$$

where λ is the power parameter and $\mathrm{GM}(s)$ the geometric mean of the statistics. Since this transformation is only possible if $s > 0$, `ABCsampler` forces the statistics to be in the interval $[1, 2]$ by the following transformation:

$$s' = \frac{s - \min(s)}{\max(s) - \min(s)} + 1. \tag{4}$$

If the optional parameter `doBoxCox` is set to 1, `ABCsampler` will perform the Box-Cox transformation prior to the linear transformation on the statistics. The values for $\min(s)$, $\max(s)$, λ and $\mathrm{GM}(s)$ have to be specified in the file defining the linear combinations. In fact, each line of this file contains these four additional columns in the same order, just after the name of the statistics, followed by the mean and standard deviation used to standardize the Box-Cox transformed statistics prior to the linear transformation. Again, if no standardization is required, just set the mean to 0 and the standard deviation to 1. If you specify a name of a file containing all values to use the Box-Cox transformation but set `doBoxCox` to 0 or do not specify `doBoxCox` at all, `ABCsampler` will interpret the columns in the file as if no Box-Cox transformation is performed, leading to strange results including four additional linear components.

2.9 Passing the observed data: the `.obs` file

Some sampling types require to evaluate the distance between the simulated and the observed data. `ABCsampler` requires the observed data to be passed via a special file called `.obs` file in the following. The name of this file is given in the `.input` file or via the command line with the parameter tag `obsName`. The file itself consists of two lines, the first containing the names of the summary statistics, and the second the corresponding values. The different columns have to be separated either by a tab or a whitespace. It has therefore exactly the same form as the file with summary statistics read by `ABCsampler` in each iteration (see Section 2.5.1 "File with computed summary statistics" on page 17).

Note that `obsName` is a mandatory parameter for any sampler type. The reason is that `ABCsampler` evaluates the simulated data and compares it to the observed data in the first iteration. That way, `ABCsampler` assures that all necessary summary statistics are calculated before launching many iterations. Also, `ABCsampler` only stores those summary statistics presented in the `.obs` file. If a standard sampling approach is launched, the actual values from the different summary statistics in the `.obs` file have no influence on the sampling process. Several files may be specified by delimiting them with a ";" (see

Section 2.7.2 "Several summary statistics computations per iteration" on page 19).

2.10 Output files

Despite writing a detailed `.log` file containing useful runtime information and potential error messages, `ABCsampler` writes a huge file containing all parameter values used to perform simulations, along with the corresponding summary statistics. The first column will give the iteration number. If present and set to 1, the parameter `addDistanceToOutputfile` controls, if the distance to the observed data is computed and added to the output file as the last column. If several simulations per iteration are performed (see Section 2.7 "Several simulations per iteration" on page 18), the parameter `separateOutputFiles` controls, if all calculated statistics from different files given by `simDataName` are written into a single file (`separateOutputFiles` 0) or not (`separateOutputFiles` 1). If separate output files are requested, each will contain the columns with the parameters used. If the all statistics are written to the same file, their names bear a prefix of the form "Obs$n$_", where n is a number corresponding to the simulation number in an iteration. Note that separate output files can not be written, if linear combinations of summary statistics are used (see Section 2.8 "Linear transformed statistics" on page 20). The reason is that the linear combinations include summary statistics from all simulations per iteration.

2.11 Sampler Types

So far we just focused on setting up a simple rejection algorithm. `ABCsampler`, however, offers currently three different sampler types:

STANDARD A standard sampling approach used to perform rejection sampling. With this sampling type, algorithms Ⓐ through Ⓓ from Figure 15 are possible. The minimal `.input` file given in Figure 3 will launch `ABCsampler` in standard sampling mode.

MCMC The MCMC without likelihood approach proposed by Marjoram et al. (2003), allowing for algorithms Ⓐ through Ⓓ from Figure 15.

PMC The population Monte Carlo approach proposed by Beaumont et al. (2009). This approach can be used just as the MCMC without likelihood approach, therefore allowing for algorithms Ⓐ through Ⓓ from Figure 15.

The preferred sampler type to use is specified with the parameter `samplerType` in the `.input` file or on the command line. The next sections will give a detailed descriptions on the specific algorithm implemented and on how to control them.

2.11.1 Launching an MCMC without Likelihoods

An introduction to the MCMC without likelihood algorithm and a detailed description of the implemented algorithm is given in Section 6.2.3 "MCMC without likelihoods" on page 55. We strongly recommend to read these sections when using this approach. Here we only outline the parameters with which an MCMC without likelihood run can be configured in `ABCsampler`:

`nbSims` controls the number of simulations to perform. Note that this number includes all simulations performed in the start-up phase (see below).

`numCaliSims` controls the number of simulations to perform as the initial calibration set. This is a mandatory parameter even if a calibration file is provided (see below). We recommend to use at least 5,000 simulations for calibration.

`tolerance` controls the tolerance level ϵ used to define δ_ϵ. Note that $0 < \epsilon \leq 1$. This is a mandatory parameter.

`rangeProp` The transition kernel implemented in `ABCsampler` is defined uniform with width φ expressed in units of standard deviations for each parameter (Wegmann and Excoffier (2008) recommended $\varphi = 0.5$). The fraction φ of the standard deviation is specified with the parameter `rangeProp`. Note that all parameters are updated in every iteration. `rangeProp` is a mandatory parameter.

`mcmcSampling` controls the interval between iterations that are printed into output files. With a `mcmcSampling` value of 1, `ABCsampler` prints every iteration, with a value of 2 every second, and so forth. It is possible to request different sampling schemes during one run by providing several values delimited by a ";". A value "1;5", for instance, requests `ABCsampler` two write two output files, one with every iteration and one with every fifth. `mcmcSampling` is a mandatory parameter. Wegmann and Excoffier (2008) used a value of 1.

`startupLength` controls the amount of iterations from the beginning, after which the chain is restarted if it did not move. This is an optional parameter, the default value is 20.

`startupAttempts` specifies the number of attempts to restart the chain. In each attempt, if the chain did not move after `startupAttempts` iterations, the chain is restarted from a new random positions among the $n\epsilon$ positions accepted from the initial calibration simulations. `startupAttempts` is an optional parameter, the default value is 100.

`stopIfStartupFailed` changes the behavior of `ABCsampler` after `startupAttempts` failed. If it is set to 1, the program is aborted. If it is set to 0, the chain will proceed until the amount of simulations specified with the parameter `nbSims`, including those in the burn in phase, are performed. `stopIfStartupFailed` is an optional parameter, the default value is 0.

`runsPerParameterVector` sets the number of simulations per iteration. If `runsPerParameterVector` > 1, several simulations with the same model parameter values will be launched per iteration. The simulated summary statistics of all these simulations are outputted in the case of a STANDARD sampling. In case of an MCMC without likelihoods (or PMC) run, the decision to accept or reject the values of the model parameters is based on the average distance of the different simulations. The summary statistics of the last run will be outputted only. Note that if `runsPerParameterVector` > 1 the chain will converge to a distribution which is different from $\pi(\theta|\text{dist}(\mathbf{s}, \mathbf{s}_{obs}) < \delta_\epsilon)$. `runsPerParameterVector` is an optional parameter and the default (and recommended) value is 1.

Calibration File The simulations used to initially calibrate an MCMC without likelihood chain may also be provided to `ABCsampler` via a file. The name of this file is specified in the `.input` file or via the command line using the parameter `calName`. The file itself has a similar structure like the output file of `ABCsampler`: each line contains the values of the model parameters and the values of the statistics computed on the output of a simulation with these model parameter values. The first line contains the names of the model parameters and statistics. Note that the names have to correspond to those defined either in the `.est` file (see Section 2.3 "Defining prior distributions - `.est` File" on page 11) or in the `.obs` file (see Section 2.9 "Passing the observed data: the `.obs` file" on page 21). Of course the calibration file may contain additional columns which will be ignored by `ABCsampler`.

The number of simulations to read and use for the calibration is defined by the parameter `numCaliSims`. Note that an error will be thrown, if the provided file contains less simulations than indicated by `numCaliSims`.

2.11.2 Launching a Population Monte Carlo algorithm

An introduction to population Monte Carlo (PMC) and a detailed description of the implemented algorithm is given in Section 6.2.4 "Population Monte Carlo" on page 57. We recommend to study this section when using a PMC approach. Here we only outline the parameters with which the PMC algorithm implemented in `ABCsampler` is controlled:

`numInterations` controls the number of iterations to be performed.

`sampleSize` controls the number of simulations that are performed in one iteration.

`lastSampleSize` specifies the number of simulations that are performed in the last iteration. In most cases this is the pool of simulations used for further analysis. This is an optional parameter. If it is not given, `ABCsampler` will generate the same number of simulations in the last as in previous iterations.

`tolerance` controls the tolerance level ϵ_t used to define δ_{ϵ_t} in each iteration. The number of parameter vectors used to generate the next iteration $t+1$ are the $\epsilon_t N$ parameter vectors with the closest associated distances of the N parameter vectors of the current iteration t, therefore all $\boldsymbol{\theta}_{i,t}$ with $\text{dist}(\mathbf{s}, \mathbf{s}_{obs}) < \delta_{\epsilon_t}$. Note that $0 < \epsilon \leq 1$. This is a mandatory parameter.

`rangeProp` The transition kernel proposed by Beaumont et al. (2009) and implemented in `ABCsampler` is defined as a multivariate normal with width τ_t^2 expressed in units of the variance-covariance matrix σ_t^2 of the accepted parameter vectors. The τ_t^2 used by `ABCsampler` is simply σ_t^2 multiplied by the parameter `rangeProp`. Since Beaumont et al. (2009) showed that $\tau_t^2 = 2\sigma_t^2$ is an optimal choice, the default value of `rangeProp` is 2.

`numCaliSims` controls the number of simulations to perform in the first iteration and is a mandatory parameter. It may be desired to use a different number of simulations in the first iteration, since these simulations are also used to find the mean and standard deviation of the statistics under the prior to calibrate the distance calculation. Note that the number of accepted parameter vectors in the first iteration does not differ from the following generations, it will be $\epsilon_t N$ (see above). `ABCsampler` will throw an error if `numCaliSims` is smaller than $\epsilon_t N$. The simulations of the first iterations, the calibration simulations, may also be provided as a file with the `calName` (see below).

`calName` specifies that `ABCsampler` will attempt to read the simulations of the first iteration from a file named `calName`. `ABCsampler` will read as many simulations from the file as indicated by the parameter `numCaliSims` and throw an error if the file contains less simulations. Note that this also offers the possibility to prolong a previous PMC run since an output file of any iteration can be used as a calibration file for a new run. The reason is that any iteration of a PMC run generates a sample of parameter vectors that corresponds to a sample acquired with a rejection sampling algorithm with the associated distance threshold δ_{ϵ_t}. Note, however, that if no linear transformations are used, `ABCsampler` will standardize the statistics using the information contained in this file, which will potentially change the contribution of some statistics to the distance, if the output of a previous iteration is used as a new calibration file. The parameter `calName` is an optional parameter.

Note that there are a couple of parameters available for a PMC run that work just as for an MCMC without likelihood run. Check the table in Section 2.12 "All Available Parameters for `ABCsampler`" on page 25 for a complete list.

Output of a Population Monte Carlo run Different to other sampler types (see Section 2.11 "Sampler Types" on page 22) a PMC run does not output a single result file but a file for every iteration. Any of these files is a valid output of the PMC algorithm and corresponds to a sample of a simple rejection algorithm with the same threshold distance and may be used for further analysis such as using a post–sampling adjustment (see Section 6.3 "Post–sampling Adjustment" on page 58). The threshold distances δ_{ϵ_t} of the different iterations are given in the `.log` file (see Section 2.10 "Output files" on page 22). Note that these threshold distances may converge very quickly such that sometimes slightly larger distances are used than in the previous iteration. This indicates that the algorithm is not able to generate samples closer to s_{obs} without additional effort (more simulations per iteration). This may, however, not be very economic. We rather suggest in this case to generate more simulations in the last iteration and use a simple rejection approach in the end, just as proposed by Wegmann and Excoffier (2008) for the MCMC without likelihood approach. `ABCsampler` offers the possibility to generate a larger sample in the last iteration with the parameter `lastSampleSize` (see above).

2.12 All Available Parameters for `ABCsampler`

The following table lists all available parameters for `ABCsampler`. The last two columns indicate whether a parameter is required (R) or optional (O) for the two sampler types, standard (S) and MCMC (M). Note that the indicated default values are used if the parameter is not passed to `ABCestimator`.

Parameter	Description and possible values	see...	SMP
`addDistanceToOutputfile`	Controls if the distance to the observed data is computed and added to the output file for every simulation. 0 (default) or 1	2.10, p. 22	OOO
`addToSeed`	When initializing the random generator, this value is added to the seed obtained from the current time. If a large set of jobs is launched on a grid, all jobs might have the same seed. By adding a value such as the job number, this can be circumvented. Any Integer (default is 0)	5.5, p. 51	OOO
`calName`	The name of the calibration file. Filename[1]	2.11.1, p. 23	– OO
`doBoxCox`	Controls if the statistics are transformed via Box–Cox when using linear transformations. 0 (default) or 1	2.8, p. 20	OOO
`estName`	Name of the `.est`-file, the file with the prior definitions. Filename[1]	2.3, p. 11	RRR
`lastSampleSize`	The number of simulations to perform in the last iteration of a PMC run. Integer	2.11.2, p. 23	– – O
`launchAfterSim`	Name of the script or program launched just after the simulation program was called. Executable[3]	2.6, p. 17	OOO
`launchAfterSimParam`	The parameters passed to the scripted or program launched just after the simulation program was called. parameter string[2]	2.6, p. 17	OOO
`launchAfterSS`	Name of the script or program launched just after the program calculating summary statistics has been called. Executable[3]	2.6, p. 17	OOO
`launchAfterSSParam`	The parameters passed to the scripted or program launched just after the program calculating summary statistics has been called. Parameter string[2]	2.6, p. 17	OOO
`launchBeforeSim`	Name of the script or program launched just before the simulation program will be called. Executable[3]	2.6, p. 17	OOO
`launchBeforeSimParam`	The parameters passed to the scripted or program launched just before the simulation program will be called. Parameter string[2]	2.6, p. 17	OOO

[1]The name of an existing file.

[2]The different values to pass are delimited by a "#". Example: "1#a string#0.5". See individual explanations.

[3]An executable file, may be a program or a script. On a Unix system, remember to grant executing permissions of the file to `ABCsampler`.

Parameter	Description and possible values	see...	SMP
`launchBeforeSS`	Name of the script or program launched just after the simulation program has been called. Executable[3]	2.6, p. 17	OOO
`launchBeforeSSParam`	The parameters passed to the scripted or program launched just after simulation program has been called. Parameter string[2]	2.6, p. 17	OOO
`linearCombName`	The name of the file where linear combinations of the summary statistics are defined. Filename[1]	2.8, p. 20	OOO
`mcmcSampling`	Specifies the sampling interval of an MCMC chain. Integer, default is 1	2.11.1, p. 22	– O –
`nbSims`	Number of Simulations to perform, includes start-up in the case of an MCMC. Integer	2.11.1, p. 22	RR –
`numCaliSims`	Number of simulations used to calibrate an MCMC Chain. This parameter is also required if a calibration file is specified. Integer	2.11.1, p. 22	– RR
`numInterations`	The number of iterations in a PMC run. Integer	2.11.2, p. 23	– – R
`obsName`	Name of the file with the observed statistics. Filename[1]	2.9, p. 21	RRR
`rangeProp`	A Parameter controlling the width of the proposal kernel in an MCMC chain. Float	2.11.1, p. 22	– RR
`runsPerParameterVector`	Controls the amount of iterations per parameter vector. Integer, default is 1	2.11.1, p. 22	OOO
`samplerType`	Controls which type of sampler to be launched. “standard” or “MCMC”	2.11, p. 22	RRR
`sampleSize`	The number of simulations to perform per iteration in a PMC run. Integer	2.11.2, p. 23	– – R
`separateOutputFiles`	If several simulations are performed per iteration, this parameter controls if the output is written to individual files. 0 (default) or 1	2.7, p. 18	OOO
`simDataName`	Name of the file with simulated data. Filename[1]	2.5, p. 16	RRR
`simInputName`	Name of the input file for the simulation program. Filename[1]	2.4, p. 13	OOO
`simParam`	Parameters to be passed to the simulation program. parameter string[2]	2.4, p. 13	RRR
`simulationProgram`	Name of the simulation program. Executable[3]	2.4, p. 13	RRR
`startupAttempts`	Number times an MCMC chain is restarted if it did not move within `startupLength` steps. Integer, default is 100	2.11.1, p. 22	– O –
`startupLength`	Number of steps an MCMC chain is given time to move before it is restarted. Integer, default is 20	2.11.1, p. 22	– O –
`stdLinearCombForDist`	Controls if the linear combinations are standardized to calculate the distance in an MCMC chain or a PMC run. 0 (default) or 1	2.11.1, p. 22	– OO

Parameter	Description and possible values	see...	SMP
`stopIfSartupFailed`	Controls if the sampler proceeds if the chain did not not move after an MCMC chain has been restarted `startupAttempts` times. 0 (default) or 1	2.11.1, p. 22	– R –
`sumStatFile`	name of the file with the computed summary statistics. The default value is "summary_stats_temp.txt".Filename[1]	2.5.1, p. 17	OOO
`sumStatParam`	Parameters to be passed to the program calculating summary statistics. Parameter string[2]	2.5, p. 16	OOO
`sumStatProgram`	Name of the program calculating summary statistics. Executable[3]	2.5, p. 16	OOO
`tolerance`	The tolerance level to be used in an MCMC chain or a PMC run. Float	2.11.1, p. 22	– RR

2.13 Possible error Messages

Error messages are written to the command line and to the `.log` file. Here you find a hopefully complete list of possible error messages and some suggestions on what could be changed to make `ABCsampler` work.

Error when executing '...'!
An error occurred when the specified program or script was launched. Note that `ABCsampler` was able to locate the file before its first execution. Most likely `ABCsampler` does not have the permission to execute the script or program. If the operating system reports an error of the script or program, this error message may also occur. It is sometimes useful to run the desired program or script with an already parsed input file (see Section 2.4 "Using different Simulation Programs" on page 13) to see what the exact error is. Note that `ABCsampler` can not report errors of called programs.

Calibration file '...' could not be opened!
The calibration file specified with the parameter `calName` was not found by `ABCsampler`. Most likely, the file is not present in the working directory of `ABCsampler`. See Section 2.11.1 "Calibration File" on page 23.

Calling program '...': not able to create fork!
The operating system did not allow `ABCsampler` to launch the specified program. This may be the case, if there are not enough available resources on the machine.

Column name '...' in the file containing the computed summary statistics '...' missing!
A summary statistics specified in the `.obs` file is missing in the file containing the computed summary statistics. Often the program or script calculating the summary statistics is launched with missing arguments. If the program or script computing the summary statistics fails and just creates an empty file this may also result in this error. Note that the names in the `.obs` file should not bear prefixes. So remove them if you use pseudo observed data sets generated with `ABCsampler`. See Section 2.10 "Output files" on page 22.

Hyperprior '...': stdev of the log normal distribution ≤ 0!
If you use the prior of a model parameter as hyper prior for a log normal distribution, make sure the value used as the standard deviation is larger than zero. See Section 2.4.1 "Hyper Priors" on page 14.

Hyperprior '...': stdev of the log normal distribution with parameters in log scale ≤ 0!
If you use the prior of a model parameter as hyper prior for a log normal distribution with parameters in log scale, make sure the value used as the standard deviation is larger than zero. See Section 2.4.1 "Hyper Priors" on page 14.

Hyperprior: mean of the beta distribution > 1!
If you use the prior of a model parameter as hyper prior for a beta distribution, make sure its value, which is used as the mean of the beta distribution, does not exceed one. See Section 2.4.1 "Hyper Priors" on page 14.

File with simulations '. . . ' is too short!
The calibration file, specified with `calName` contains less lines than is required by the parameter `numCaliSims`. See Section 2.11.1 "Calibration File" on page 23.

Given the defined threshold, no simulations is retained!
The product of the `sampleSize` and the `tolerance` specified for a PMC run is smaller than one. Either increase the `sampleSize` or the `tolerance`. See Section 2.11.2 "Launching a Population Monte Carlo algorithm" on page 23.

Given the defined threshold, all simulations are retained!
The product of the `sampleSize` and the `tolerance` specified for a PMC run is larger than the `sampleSize`. You should decrease the `tolerance`. See Section 2.11.2 "Launching a Population Monte Carlo algorithm" on page 23.

Less calibration simulations than simulations to retain in the first iteration!
The product of the `sampleSize` and the `tolerance` set the number of parameter vectors (simulations) to accept in each iteration. You specified a smaller number of simulations to be used in the first iteration. Increase the parameter `numCaliSims`. See Section 2.11.2 "Launching a Population Monte Carlo algorithm" on page 23.

Linear transformation can not be used, if several output files are written!
Most likely you forgot to remove the request to write separate output files. See Section 2.10 "Output files" on page 22.

Log normal prior '. . . ' initialized with min < 0!
Log normal prior distributions are only defined above zero. See Section 2.3 "Defining prior distributions - `.est` File" on page 11.

Log normal prior '. . . ' initialized with stdev ≤ 0!
The standard deviation of a log normal distributions has to be larger than zero. See Section 2.3 "Defining prior distributions - `.est` File" on page 11.

Normal prior '. . . ' initialized with stdev ≤ 0!
The standard deviation of a normal distributions has to be larger than zero. See Section 2.3 "Defining prior distributions - `.est` File" on page 11.

Prior '. . . ' initialized with min $>$ max!
The minimum indicated for any prior distribution may not exceed the indicated maximum. See Section 2.3 "Defining prior distributions - `.est` File" on page 11.

Problems reading complex parameter (Line: '. . . ')!
`ABCsampler` was unable to interpret this line of the `[COMPLEX PARAMETERS]` section of the `.est` file. See Section 2.3.3 "`[COMPLEX PARAMETERS]`" on page 12.

Problems reading prior (Line: '. . . ')!
`ABCsampler` was unable to interpret this line of the `[PARAMETERS]` section of the `.est` file. See Section 2.3.1 "`[PARAMETERS]`" on page 11.

Problems reading rule (Line: '. . . ')!
`ABCsampler` was unable to interpret this line of the `[RULES]` section of the `.est` file. See Section 2.3.2 "`[RULES]`" on page 12.

Problems reading rule: parameter '. . . ' does not exist!
A parameter used in the `[RULES]` was defined in the `[PARAMETERS]` section of the `.est` file. See Section 2.3.2 "`[RULES]`" on page 12.

Problems solving equation '. . . ' (closing bracket missing)!
`ABCsampler` was unable to calculate this equation defined in the `.est` file. Most likely, a bracket is missing. See Section 2.3 "Defining prior distributions - `.est` File" on page 11.

Problems solving equation '. . . ': devision by zero!
Solving this equation (defined in the `.est`–file), a division by zero was attempted. See Section 2.3.3 "`[COMPLEX PARAMETERS]`" on page 12.

Problems solving equation '. . . ': function '. . . ' not known)!
The used function in this equation defined in the `.est` file is unknown to `ABCsampler`. See Section 2.3.3 "`[COMPLEX PARAMETERS]`" on page 12 for a complete list of available functions.

Problems solving equation '. . . ': no known sign used!
Either there is an operator missing in this equation defined in the `.est`, or the used sign is unknown to `ABCsampler`. See Section 2.3 "Defining prior distributions - `.est` File" on page 11.

Problems solving the Cholesky decomposition of Sigma!
In a PMC run, the variance–covariance matrix of the retained parameters (Sigma) has to be Cholesky–decomposed in order to generate the multivariate normal transition kernel. This decomposition is only possible if Sigma is a symmetric, positive-definite matrix. The latter condition is hard to hold, if the number of retained simulations is very small. Try to retain more simulations in your PMC run (see Section 2.11.2 "Launching a Population Monte Carlo algorithm" on page 23). See Section 2.3 "Defining prior distributions - `.est` File" on page 11.

Section `[PARAMETERS]` is missing in the .est file '. . . '!
The section `[PARAMETERS]` is the only mandatory section of the `.est` file. See Section 2.3 "Defining prior distributions - `.est` File" on page 11.

The .est file '. . . ' could not be opened!
The file with the prior definitions, the `.est` file, was not found by `ABCsampler`. Most likely, the file is not present in the working directory of `ABCsampler`. See Section 2.3 "Defining prior distributions - `.est` File" on page 11.

The .obs file '. . . ' could not be opened!
The file containing the observed data and specified by `obsName` can not be read by `ABCsampler`. Most likely, the file is not present in the working directory of `ABCsampler`. See Section 2.9 "Passing the observed data: the `.obs` file" on page 21.

The data column '. . . ' is missing in the calibration File!
Values for all parameters and complex parameters specified in the `.est` file and for all statistics defined in the `.obs` file have to be present in the calibration file. Apparently, a summary statistics used in the `.obs` file is missing in the provided calibration file. See Section 2.11.1 "Calibration File" on page 23.

The file containing the computed summary statistics '. . . ' could not be opened!
`ABCsampler` requires a file from which the computed summary statistics are read. This file is written by the program calculating the summary statistics or a script. Most likely this program or script failed to write the file or wrote the summary statistics to a file with a different filename from the one specified with `sumStatFile`. If you run the program calculating summary statistics several times per iteration, check the `.log` file to see in which iteration the error occurred. The `.log` also informs you, if the error occurred while initializing `ABCsampler` or during the run. The latter would most likely correspond to a case where the simulated data cannot be handled by the program or script calculating the summary statistics. This may be, if the simulation program fails to simulate data under the current values of the model parameters. Some simulation programs, including `simcoal2`, return zero or "false" in such a case[2], which is handled by `ABCsampler` by printing the used model parameter values to the `.log` file. See Section 2.5.1 "File with computed summary statistics" on page 17.

The file containing the simulated data '. . . ' has not been found!
One of the files indicated by `simDataName` was not found by `ABCsampler`. Apparently the simulation program did not write this file. Either the simulation program failed or the resulting data is written to a file with a different name. See Section 2.5 "Using different Summary Statistics Programs" on page 16.

The .input file '. . . ' could not be opened!
`ABCsampler` requires a valid `.input` file as its first argument. Most likely, the file is not present in the working directory of `ABCsampler`. See Section 2.2 "`.input` File" on page 11.

The Linear–Combination file '. . . ' can not be read!
The file with the definitions of the linear combinations of summary statistics to be used, defined by `linearCombName`, can not be read. Most likely, the file is not present in the working directory of `ABCsampler`. See Section 2.8 "Linear transformed statistics" on page 20.

[2]Note that `simcoal2` does not report any error in the `.par` file. It is very often useful to run `simcoal2` with the `.par` file generated by sampler by hand to check if everything is fine.

The number of .obs files and files with simulated data are unequal!
If several simulations per iteration are performed, for each file with simulated data, specified by `simDataName`, a corresponding file with observed summary statistics, specified by `obsName`, has to be defined. See Section 2.7.2 “Several summary statistics computations per iteration” on page 19.

The number of simulation programs and the number of argument lists for the simulation program do not match!
The number of simulation programs given by `simulationProgram` has to match the number of argument lists provided by `simParam` for these programs. Only exception: A single simulation program may be given, but the number of argument lists matches the number of simulation program inputfiles given by `simInputName` See Section 2.7.1 “Several calls of the simulation program per iteration” on page 19.

The parameter ’. . . ’ is not defined in the .input file!
A mandatory parameter for this sampling type was not specified in the `.input` file nor passed via the command line. See Section 2.1 “Launching `ABCsampler`” on page 11 for a description on how to pass parameters to `ABCsampler` and Section 2.12 “All Available Parameters for `ABCsampler`” on page 25 for a complete list of all parameters.

The parameter column ’. . . ’ is missing in the calibration file!
Values for all parameters and complex parameters specified in the `.est` file and for all statistics defined in the `.obs` file have to be present in the calibration file. Apparently, a parameter or complex parameter form the `.est` file is missing in the provided calibration file. See Section 2.11.1 “Calibration File” on page 23.

The simulation program input file ’. . . ’ could not be read!
One of the files specified with `simInputName` was not found by `ABCsampler`. Most likely, the file is not present in the working directory of `ABCsampler`. See Section 2.4 “Using different Simulation Programs” on page 13.

The summary statistics ’. . . ’ is missing in the simulated file, but required by the Linear–Combination file ’. . . ’!
A summary statistics used to compute linear combinations as specified in the file given by `linearCombName` was not simulated. Most likely you did not include this statistics in the `.obs` file. Note that this error will also occur if you included a model parameter as a summary statistics in the file defining the Linear–Combinations. See Section 2.9 “Passing the observed data: the `.obs` file” on page 21.

Unequal number of columns among the lines in the Linear-Combination-File ’. . . ’!
Please check the format of the file with the definitions of the linear combinations of summary statistics to be used, defined by `linearCombName`. See Section 2.8 “Linear transformed statistics” on page 20.

Unequal number of files with simulated data sets and parameter specifications for the script/program to launch after the calculation of summary statistics!
If several simulations per iteration are performed, the number of argument lists for the script launched before the calculation of the summary statistics has to match the number of files containing simulated data (given by `simDataName`). Only exception: a single argument list is provided and used for all files with simulate data. See Section 2.7.2 “Several summary statistics computations per iteration” on page 19.

Unequal number of files with simulated data sets and parameter specifications for the script/program to launch before the calculation of summary statistics!
If several simulations per iteration are performed, the number of argument lists for the script launched before the calculation of the summary statistics has to match the number of files containing simulated data (given by `simDataName`). Only exception: a single argument list is provided and used for all files with simulate data. See Section 2.7.2 “Several summary statistics computations per iteration” on page 19.

Unequal number of input files and argument lists for the simulation program!
The number of simulation programs given by `simulationProgram` and the number of argu-

ment lists provided by `simParam` have to match the number of simulation program inputfiles given by `simInputName`. Only exception: a single simulation program may be given, but the number of argument lists has to match the number of simulation program inputfiles given by `simInputName`. See Section 2.7.1 "Several calls of the simulation program per iteration" on page 19.

Unequal number of input files and parameter specifications for the script/program to launch after the calculation of summary statistics!
The number argument lists for the script/program to be launched after the calculation of summary statistics (given by `launchAfterSSParam`) has to match the number of files containing simulated data (given by `simDataName`). Only exception: a single argument list may be given, which is then used for all files containing simulated data. See Section 2.7.2 "Several summary statistics computations per iteration" on page 19.

Unequal number of input files and parameter specifications for the script/program to launch before the calculation of summary statistics!
The number argument lists for the script/program to be launched before the calculation of summary statistics (given by `launchBeforeSSParam`) have to match the number of files containing simulated data (given by `simDataName`). Only exception: a single argument list may be given, which is then used for all files containing simulated data. See Section 2.7.2 "Several summary statistics computations per iteration" on page 19.

Unknown prior type (line: '...')!
The prior type specified on this line of the `.est` file is unknown. Make sure to have a single whitespace OR tab between the name of the model parameter and the prior type tag. See Section 2.3.1 "`[PARAMETERS]`" on page 11.

Unknown samplerType '...'!
You specified an unknown `samplerType`. See Section 2.11 "Sampler Types" on page 22.

Missing first argument, the name of an .input file!
`ABCsampler` was launched without any arguments. Note that `ABCsampler` requires at least the name of a valid `.input` file as its first argument. See Section 2.1 "Launching `ABCsampler`" on page 11.

Wrong number of programs to calculate summary statistics or defined argument lists for these programs!
If several simulations per iteration are performed, the number of programs to calculate summary statistics has to match the number of files with simulated data defined by `simDataName`. Only exception: a single program is used for all files with simulated data. Also, the number of programs to calculate summary statistics has to match the number of argument lists provided! If only one program is provided, still several argument lists are possible, as long as they match the number of files with simulated data. See Section 2.7.2 "Several summary statistics computations per iteration" on page 19.

2.14 Compilation

`ABCsampler` was compiled and tested with the `GCC 4.3` compiler on a Linux system using a standard makefile. It was also successfully compiled with GCC on Windows using the MinGW package. Since file handling and program execution is different between the two operating systems, not all functionalities are available if compiled with Borland. This is namely the use of scripts / programs as described in Section 2.6 "Modifying files at every iteration with bash scripts or programs (optional)" on page 17. To use the whole functionality use the compilation variable `_GCC_` with the `GCC` compiler on a Linux system. We were not able to compile `ABCsampler` sucessfully under windows using thsi compilation variable.

3 Using `ABCestimator`

The program `ABCestimator` is used to perform a post–sampling regression adjustment on a data set. Currently, two estimation types exist: the ABC-GLM approach introduced by Leuenberger and Wegmann (2009), and a slightly modified approach, where the independence of individual measures of a single model are explicitly taken into account (Thalmann et al. 2009). Under the assumption of a good fit of the GLM to the data within the ϵ ball, both approaches should lead to the true posterior distribution, as was shown by Leuenberger and Wegmann (2009) and is described in Section 6.3.2 "ABC-GLM" on page 60. See below for a detailed description on how to use `ABCestimator` to perform these regression adjustments.

3.1 Launching `ABCestimator`

Just as `ABCsampler`, `ABCestimator` is a command line program launched with one mandatory argument, the name of the `.input` file. All necessary parameters are specified in the `.input`–file (see below). Suppose the inputfile is named `name.input`, the program has to be launched as follows:

```
$ ABCestimator name.input
```

Additional parameters may be passed on the command line. In that case, parameters given in the `.input` file will be overwritten. The required syntax is as follows:

```
$ ABCestimator name.input parameter=value
```

where `parameter` is the name of a parameter and `value` is the corresponding value. For instance,

```
$ ABCestimator name.input threshold=0.1
```

launches `ABCestimator` with a threshold of 0.1 to be used.

There is no restriction for the number of parameters to be passed that way. Note, however, that the program always requires the name of an existing `.input` file as the first parameter. `ABCestimator` writes all parameters used, the progress of the estimation and important messages directly to the standard output. The main output of `ABCestimator` is a file with the poster estimates, along with some additional files containing additional information. See Section 3.10 "Output" on page 36 for e detailed description of the output of `ABCestimator`.

3.2 `.input` File

An `.input` file is the most convenient way to specify parameters for `ABCestimator`. One reason is that the parameters used are stored in a reusable fashion. Basically, an `.input` file is a simple collection of pairs of parameter tags and corresponding values, enriched with comments. It has exactly the same format as the `.input` file of `ABCsampler` (see Section 2.2 "`.input` File" on page 11): each parameter–value pair has to be at the beginning of a new line but the order is of no importance. Comments may either be added after a parameter–value pair or given on a line on their own. Note, however, that all comments must be proceeded with a double slash "//". Empty lines may be present anywhere within the file. A minimal `.input` file for `ABCestimator` is given in Figure 7. A complete list of all available parameter tags is given in Section 3.11 "All available Parameter for `ABCestimator`" on page 37.

3.3 Observed Data

The observed data, that is, the summary statistics calculated on the observed data sets, is passed to `ABCestimator` in a text file, the name of which is specified by `obsName` in the `.input` file or via the command line. The file itself contains a header line with the names of the summary statistics delimited by a tab or a whitespace, followed by individual data sets, each on a single line. Each data set must consist of all summary statistics in the same order, delimited by the same character as the names given in the header line. The file has thefore exactly the same format as the `.obs`–file passed to `ABCsampler` (see Section 2.9 "Passing the observed data: the `.obs` file" on page 21), except that several data sets may be provided. The handling of the different data sets depends on the estimation type used (see below).

```
//inputfile for the program ABCestimator
estimationType standard simFile allsimulations.txt
obsFile observedData.obs
params 1-22
//rejection
numRetained 5000
//parameters for posterior estimation
diracPeakWidth 0.01
posteriorDensityPoints 200
stadardizeStats 1
writeRetained 1
```

Figure 7. A *ABCestimator* input file *This is an example of a very simple .input file for the program ABCestimator. Each line represents a pair of a parameter and a corresponding value. A complete list of all parameters is given in Section 3.11 "All available Parameter for ABCestimator" on page 37. Note that comments, indicated with //, are possible on whole lines or after the declaration of a parameter.*

3.4 Simulated Data

As explained in detail in Section 6.3.2 "ABC-GLM" on page 60, any ABC estimations relies on a huge set of simulated data sets with parameter values drawn by an ABC sampler and corresponding summary statistics. `ABCestimator` performs a post–sampling regression adjustment on such a data matrix, which is provided to `ABCestimator` as a huge text file. The name of the file is specified with the parameter `simName`, either in the `.input` file or via the command line. Each individual line of this file contains a single simulated data set consting of model parameters of the model and corresponding summary statistics obtained through simulations. Model parameters and statistics are organized in columns delimited by a tab or a whitespace. While the order of the columns may be arbitrary, all lines have to follow this order strictly. The first line of the file is a header line containing the names of the columns. These names must not contain spaces nor tabs. The length of the file is unlimited, however, the actual number of lines that can be handled by `ABCestimator` depends on available memory[3]. Several millions of simulations should not be a problem. If only a sub set of the simulation should be used, this can be specified with the parameter `maxReadSims`. `ABCestimator` will then only use the first `maxReadSims` simulations of the file. If the file is shorter, this parameter is ignored.

`ABCestimator` matches the names given in the header line of the `.obs` file with the names provided in the file with the simulated data set. Missing columns in the file with the simulated data set is reported by `ABCestimator`. Unused columns with summary statistics are ignored. Columns are interpreted as model parameters according to the definition given by the parameter `params`. The value of the parameter `params` is a string giving the numbers of the columns to be read as model parameters and for which estimation are performed. The string may contain numbers of single columns or ranges, delimited by ",". Example: `params 1-4,8,10-12,14`. If the indicated column is also used as a summary statistics `ABCestimator` reports an error; unused columns are ignored.

3.5 Estimation Types

The preferred estimation type to use is specified with the parameter `estimationType` in the `.input` file or on the command line. Currently a single estimation type is implemented in `ABCsampler` and chosen by the following tag:

standard performs a post–sampling regression adjustment just as described in Leuenberger and Wegmann (2009) and introduced in Section 6.3.2 "ABC-GLM" on page 60.

[3]The implementation chosen reads the whole file in the memory. While this method may not be appropriate if extremely large data sets are considered (only the distances could be stored in memory), it speeds up the estimation process if several estimations are performed at once.

3.5.1 Standard Estimation

As mentioned in Section 6.3 “Post–sampling Adjustment” on page 58 and described in detail in Leuenberger and Wegmann (2009), the aim of the post sampling regression adjustment performed by `ABCestimator` is to describe the relationship between model parameters and summary statistics with a simple statistical model. While this model may not be realistic on the whole prior distribution, it may fit quite well locally around the observed data. `ABCestimator` proceeds therefore by a first rejection step and then fitting a statistical model to the retained values, which is basically an estimate of the likelihood of the truncated model $\mathcal{M}_\epsilon(\mathbf{s}_{obs})$, where $\mathbf{s}_{obs}$ is the vector of observed summary statistics. As was shown recently (Leuenberger and Wegmann 2009), the true posterior of the initial model $\mathcal{M}$ is exactly equal to the posterior distribution of the truncated model $\mathcal{M}_\epsilon(\mathbf{s}_{obs})$ given the truncated prior $\pi_\epsilon(\boldsymbol{\theta})$ (see Section 6.3 “Post–sampling Adjustment” on page 58 for a detailed description). If a standard estimation is launched, `ABCestimator` repeats the estimation steps for each data set provided in the `.obs`–file (see Section 3.3 “Observed Data” on page 32). Since the statistical model is (in most cases) at best a good approximation of the truncated model $\mathcal{M}_\epsilon(\mathbf{s}_{obs})$, the posterior remains approximative.

The details of the different estimations steps are given in Section 6.3.2 “ABC-GLM” on page 60 and Leuenberger and Wegmann (2009). Here we just describe how to configure a standard estimation approach.

Rejection The hope is that the statistical model fits better to the data, if rejection is stringent. On the other hand, the accuracy of the estimation of the truncated prior $\pi_\epsilon(\boldsymbol{\theta})$ requires a larger sample of simulations, such that a tradeoff has to be found. The rejection step is regulated by the two parameters `numRetained` and `tolerance`. The two parameters stand for a different rejection philosophy: while the exact distance below which simulations are accepted is given with `tolerance`, the parameter `numRetained` sets the tolerance indirectly by specifying the number of closest simulations to retain. If both are defined, the more stringent will be effective. If none is defined, all simulations will be used for model parameter estimation. If the parameter `writeRetained` is defined and set to 1, `ABCestimator` writes two files: one containing the parameter and statistics of the retained simulations and one with the smoothed parameter distribution of the retained simulations (see Section 3.10 “Output” on page 36).

The rejection step as implemented in `ABCestimator` depends on the Euclidean distance between the simulated and observed data. As will be discussed in detail in Section 5.2 “Choosing Summary Statistics” on page 44, the choice of summary statistics is an important issue with a large influence on the distance. The parameter `standardizeStats` specifies whether (1) or not (0) the statistics are standardized before the distance is calculated.

Posterior estimation A standard estimation follows exactly the formulation of Leuenberger and Wegmann (2009), which is outlined in Section 6.3.2 “Details of the Posterior Estimation” on page 60. `ABCestimator` only estimates and reports the marginal posterior density of the parameter θ_k defined by

$$\pi(\theta_k|\mathbf{s}) = \int_{\mathbb{R}^{m-1}} \pi(\boldsymbol{\theta}|\mathbf{s}) d\boldsymbol{\theta}_{-k}, \qquad (5)$$

where integration is performed along all parameters except θ_k. `ABCestimator` calculates the density of the marginal posteriors on a number of equally spaced points along the range of every parameter. The number of such points is specified with the parameter `posteriorDensityPoints` with default value 100.

The posterior estimation depends on the smoothed estimate of the truncated prior from the retained samples (see Section 6.3.2 “Details of the Posterior Estimation” on page 60 for details). The smoothing is regulated by the parameter `diracPeakWidth` (corresponds to $\boldsymbol{\Sigma}_\theta$ in the formulation of Leuenberger and Wegmann (2009)). Since `ABCestimator` stamndardizes the parameters internally to the range $[0, 1]$ `diracPeakWidth`, the same `diracPeakWidth` value is used for all parameters. Too small values of `diracPeakWidth` will result in wiggly posterior curves, too large values might unduly smear out the curves. The best advice is to run the calculations with several choices for `diracPeakWidth`.

The choice of diracPeakWidth depends on the number of retained simulations: the larger the number of retained parameter values, the sharper the smaller diracPeakWidth can be chosen in order to still get a rather smooth result. If the parameter diracPeakWidth is not defined, ABCestimator uses as value of 0.001, unless the parameter numRetained is defined. In this case ABCestimator sets $\sigma_k = 1/N$, where N is the number of simulations to retain, as proposed by Leuenberger and Wegmann (2009).

3.6 Distances from different statistics

ABCestimator offers the possibility to use different statistics for the distance calculation and the estimation itself. Simply provide another file with the simulated data by the parameter distSimFile. This file has exactly the same structure as the file given by simName (see Section 3.4 "Simulated Data" on page 33) except that it contains those statistics to be used for the distance calculation. ABCestimator also needs a file with the observed values of these summary statistics, the name of which is provided with the parameter distObsFile.

3.7 Model Choice

ABCestimator allows to perform model selection via Bayes factors. For two models $\mathcal{M}_A$ and $\mathcal{M}_B$ with prior probabilities π_A and $\pi_B = 1 - \pi_A$, the Bayes factor B_{AB} in favor of model $\mathcal{M}_A$ over model $\mathcal{M}_B$ is

$$B_{AB} = \frac{f_{\mathcal{M}_A}(\mathbf{s}_{obs})}{f_{\mathcal{M}_B}(\mathbf{s}_{obs})}, \tag{6}$$

where $f_{\mathcal{M}_A}$ and $f_{\mathcal{M}_B}$ are the marginal densities of model A and B, respectively. ABCestimator prints marginal density to the standard output with the tag "marginal density". A file ("marginal_density.txt") containing the marginal densities of all observed data sets is also written. In order to calculate Bayes factors, run ABCestimator twice with the same observed data set, but different files with simulated data. These have to be generated under different models, but contain the same set of summary statistics. Then, the Bayes factor is calculated according to (6) by simply calculating the quotient of the two marginal densities obtained through the independent run of ABCestimator.

3.8 Testing Model Fit

If requested by the parameter obsPValue, a p-value for the observed data set under the estimated GLM is reported by ABCestimator. This p-value can be used to judge, if the observed data is in agreement with the retained data. Basically, ABCestimator computes the likelihood of the observed data under the GLM and compares it to the likelihood of a number of retained data sets. The reported p-value is simply the fraction of the retained simulations with a smaller or equal likelihood. If the p-value is small, it suggests that the estimated GLM does not fit the observed data well, and in turn, that the model used to generate the simulated data sets may not fit well with the observed data. Sometimes it is worth relaxing the rejection constraints to get more simulations on which the GLM is estimated. If the computations are based on many data sets the p–value is more accurately estimated, but at some computational costs. The number of retained data sets to consider is specified with the parameter obsPValue. ABCestimator will use the number of retained simulations at most. If obsPValue is chosen smaller than the number of retained simulations, ABCestimator chooses them in the same order as in the file containing all simulations (as specified by simName, see Section 3.4 "Simulated Data" on page 33).

3.9 Quantile of the true Parameter values within the Posterior Distribution

It is possible to pass the true model parameter values for the observed data sets to ABCestimator. In order to validate the chosen estimation procedure and summary statistics it is worth checking for a potential bias in the posterior distributions. A way to do this is to check for uniformity of the posterior quantiles of the true model parameter values using pseudo–observed data sets (see Section 5.3.2 "Checking for biased posteriors" on page 48). ABCestimator calculates these posterior quantiles, if a file with the true model parameter values is passed via the parameter trueParamName. The true parameter values are, for instance, the values used to generate the pseudo–observed data sets. The passed file has to be organized in columns, where each column reports the true value of one model parameter. ABCestimator will throw an error if one of these model parameters is missing in the file containing the simulations or if one of these model parameters is not requested to be estimated (see Section 3.4 "Simulated Data" on page 33). The file with the true model parameter values, on the other hand, may contain less columns than model parameters are estimated. `ABCestimator` does simply not compute the quantiles for these model parameters. The file with the true model parameters must contain as many lines as observed data sets are passed to `ABCestimator` (see Section 3.3 "Observed Data" on page 32). `ABCestimator` will then write the quantiles of the true model parameters to a file named "quantilesOfTrueParameters.txt" (see Section 3.10 "Output" on page 36).

3.10 Output

While all parameters used, the progress of the estimation and important messages are written directly to the standard output, `ABCestimator` writes several output files containing the posterior estimations or the retained simulations. All files bear a common prefix, which can be set with the parameter `outputPrefix` and which default value is "ABC_GLM_". If a standard estimation is performed and if several data sets are provided in the `.obs` file, all files written for each data set independently bear a suffix consiting of the tag "_Obs" and the number of the data set.

The file "PosteriorEstimates.txt" contains the estimates of the posterior distributions of all model parameters. It is organized in columns with two columns for each model parameter. The first, baring the name of the parameter, gives the coordinates at which the posterior density has been estimated (see Section 3.5.1 "Posterior estimation" on page 34). The second reports the posterior density at these points.

The file "PosteriorCharacteristics.txt" summarizes some important features of the posterior distributions such as mode, mean, median, several quantiles and several highest posterior density intervals. Note that the mode reported is simply the position with the highest density among the sampled positions of the smoothed posterior distribution. Quantiles are real quantiles distributed around the median. Highest posterior density intervals (HPD) are the shortest continuous interval with an integrated posterior density of a certain value. It is similar to the Bayesian credible interval.

The file "BestSimsParamStats.txt" contains the model parameters and statistics of the retained simulations, along with the computed distance of these simulations to the observed data set and the line number of the simulation in the file with the simulated data sets. This file is only written if requested with the parameter `writeRetained`.

The file "TruncatedPrior.txt" contains the smoothed parameter distribution among the retained simulations. The format of the file is similar to the file containing the posterior estimates (see above). This file is only written if requested with the parameter `writeRetained`.

The file "marginalDensity.txt" contains the marginal densities of all observed data sets.

The file "L1DistancePriorPosterior.txt" contains, for each parameter, the L_1-distance between the inferred posterior and the prior distribution. Recall that the L_1-distance of two (continuous) densities $f(\theta)$ and $g(\theta)$

$$\|f - g\|_1 = \int |f(\theta) - g(\theta)| d\theta \tag{7}$$

measures the area between the function curves. It is equal to 2 when f and g have disjoint supports and it vanishes when the functions are identical.

The file "quantilesOfTrueParameters.txt" contains the quantiles of the true model parameters within the marginal posterior distributions (see Section 3.8 "Testing Model Fit" on page 35). The first column contains the number of the observed data set, the following columns the quantiles of the different model parameters. This file is only written if the true model parameters are passed with the parameter `trueParamName` and only if a standard estimation is performed.

3.11 All available Parameter for `ABCestimator`

The following table lists all available parameters for `ABCestimator`. The last column indicates whether a parameter is required (R) or optional (O). Note that the indicated default values are used if the parameter is not passed to `ABCestimator`.

Parameter	Description and possible values	see...	SI
`diracPeakWidth`	Specifies the width of the Dirac peaks used for smoothing the marginal posterior distributions. The values are on the parameter scaled to fit within [0,1]. Large values imply smoother curves. Float, default value is 0.001 0 (default) or 1	3.5.1, p. 34	O
`distObsFile`	A file containing the observed summary statistics used to calculate the distances, if those are different from the summary statistics used to perform the regression analysis. If this parameter is defined, the parameter `distSimFile` has to be defined as well. Filename[1]	3.6, p. 35	O
`distSimFile`	A file containing the simulated summary statistics used to calculate the distances, if those are different from the summary statistics used to perform the regression analysis. If this parameter is defined, the parameter `distObsFile` has to be defined as well. Filename[1]	3.6, p. 35	O
`estimationType`	Controls the required estimation type. “standard”	3.5, p. 33	R
`maxReadSims`	Puts a limit on the maximum simulations to read from the `simName` and the `distSimFile`. Integer, default is as many simulations as present in the file.	3.4, p. 33	O
`numRetained`	The number of closest simulations to retain for the posterior estimation. Integer larger than 10, default is the total number of simulations.	3.5.1, p. 34	O
`obsName`	The name of the file containing the observed summary statistics. Filename[1]	3.3, p. 32	R
`obsPValue`	Controls whether (1) or not (0) p-values of the observed data sets are computed. 0 (default) or 1	3.10, p. 36	O
`outputPrefix`	A tag that is added before each output file written by `ABCestimator`. String, default is “ABC_GLM_”	3.10, p. 36	O
`params`	The columns in the containing model parameters for which estimates are requested. A string of the format “1,3,5-7,9”	3.4, p. 33	RR
`posteriorDensityPoints`	The number of points along the parameter axis at which the marginal posterior density is calculated. Integer, default is 100	3.5.1, p. 34	O

[1]The name of an existing file.

Parameter	Description and possible values	see...	SI
`simName`	The name of the file containing the model parameters and corresponding summary statistics of the performed simulations. These simulations are the basis on which estimates are performed. Filename[1]	3.4, p. 33	R
`stadardizeStats`	Controls whether (1) or not (0) the summary statistics are standardized before the calculation of the distances. 0 (default) or 1	3.5.1, p. 34	O
`tolerance`	The fraction of closest simulations to retain for posterior estimation. Float between 0 and 10	3.5.1, p. 34	O
`trueParamName`	The name of a file containing the true parameters used to generate the observed summary statistics. Filename[1]	3.10, p. 36	O
`writeRetained`	Whether (1) or not (0) the smoothed marginal densities of the retained simulation is computed and written to a file. 0 (default) or 1	3.10, p. 36	O

3.12 Possible error Messages

`ABCestimator` writes error messages directly to the command line. Here you find a hopefully complete list of possible error messages and some suggestions on what could be changed to make `ABCestimator` work.

Column '...' is requested as parameter and statistics!
The name of a column in the file containing the simulations is also present in the file with the observed data sets and therefore regarded as a summary statistics by `ABCestimator`. But this column is also marked as a model parameter to be estimated with the parameter `params`. You probably don't want this column to be estimated. See Section 3.4 "Simulated Data" on page 33.

File with observed statistics '...' could not be opened!
The name of the file containing the observed data sets is specified with the `obsName`. `ABCestimator` was unable to read this file. Most likely, the file is not present in the working directory of `ABCestimator`. See Section 3.3 "Observed Data" on page 32.

File with simulations '...' could not be opened!
The name of the file containing the observed data sets is specified with the `simName`. `ABCestimator` was unable to read this file. Most likely, the file is not present in the working directory of `ABCestimator`. See Section 3.4 "Simulated Data" on page 33.

File with true parameters '...' could not be opened!"
The name of the file containing the observed data sets is specified with the `trueParamName`. `ABCestimator` was unable to read this file. Most likely, the file is not present in the working directory of `ABCestimator`. See Section 3.9 "Quantile of the true Parameter values within the Posterior Distribution" on page 35.

Given the current specified tolerance / numRetained, less than 10 (only ...) simulation are retained!
`ABCestimator` requires at least 10 simulations on which any posterior estimation can be performed. In most cases it is recommended to actually use much more simulations (several hundreds up to several thousands). Consider changing the tolerance or the number of retained simulations with the parameters `tolerance` and `numRetained`, respectively. See Section 3.5.1 "Rejection" on page 34.

Input file '...' could not be opened!
`ABCestimator` requires one mandatory argument when launched, which is the name of the file containing the specifications for the estimation run. `ABCestimator` was unable to read the `.input` file which name was passed. Most likely, the file is not present in the working directory of `ABCestimator`. See Section 3.2 "`.input` File" on page 32.

Less lines in the file with true parameters'...' than observed data set!
If a file with the true model parameters is passed, it has to contain as many lines as observed data sets are passed to `ABCestimator`. Compare the number of lines of this file with the number of lines in the file containing the observed data sets (passed with the parameter `obsName`). See Section 3.9 "Quantile of the true Parameter values within the Posterior Distribution" on page 35.

More names than values in the .obs-file '...'!
At least one line of the file containing the observed summary statistics contains more values than names are present in the header line. Check the content of this file. See Section 3.3 "Observed Data" on page 32.

More names than values in the file with true parameters '...'!
At least one line of the file containing the true model parameters contains more values than names are present in the header line. Check the content of this file. See Section 3.9 "Quantile of the true Parameter values within the Posterior Distribution" on page 35.

More values than names in the .obs-file '...'!
At least one line of the file containing the observed summary statistics contains less values than names are present in the header line. Check the content of this file. See Section 3.3 "Observed Data" on page 32.

More values than names in the file with true parameters'...'!
At least one line of the file containing the true model parameters contains less values than names are present in the header line. Check the content of this file. See Section 3.9 "Quantile of the true Parameter values within the Posterior Distribution" on page 35.

Problem calculating the standard deviation of the simulated statistic '...'!
`ABCestimator` is not able to compute the standard deviation of this statistic among the simulations. Check the distribution of this statistic (for instance in R). See Section 3.4 "Simulated Data" on page 33.

Problem reading the parameter string!
The columns of the file containing the simulated data, for which `ABCestimator` calculates marginal posterior densities, are defined with the parameter `params` in a specific format. The value of this parameter does not follow the format, please check it. See Section 3.4 "Simulated Data" on page 33.

Some statistics is/are missing in the simulation file!
There is at least one summary statistics present in the file with the observed data set, but missing in the file containing the simulations. See Section 3.4 "Simulated Data" on page 33.

The number of observed data sets differs between the file used for estimation and the one used to calculate distances!
You specified a file with observed summary statistics to be used to calculate the distances. This file, however, does not contain the same number of lines as the file with the observed data sets passed with the parameter `obsName`. Check the content of these files. See Section 3.6 "Distances from different statistics" on page 35.

The number of simulations to retain is set to a value <10!
`ABCestimator` requires at least 10 simulations on which any posterior estimation can be performed. In most cases it is recommended to actually use much more simulations (several hundreds up to several thousands). Consider changing the number of retained simulations with the parameter `numRetained`. See Section 3.5.1 "Rejection" on page 34.

The parameter '...' is not defined in the inputfile!
A required parameter for `ABCestimator` is missing in the `.input` file. Please add this parameter. See Section 3.2 "`.input` File" on page 32.

The parameter '...' is not estimated, but requested from the file with true parameter values!
The file containing the true model parameters contains a column, which is either absent in the file containing the simulations or this model parameter is not requested to be estimated with the parameter `params`. See Section 3.9 "Quantile of the true Parameter values within the Posterior Distribution" on page 35.

The statistics '. . . ' is monomorphic among the retained simulations!
`ABCestimator` can not perform the regression adjustment, if some statistics are monomorphic, which means that all retained simulations have the same value for this statistics. Consider either removing this statistics (simply drop it from the file with the observed data sets) or to retain more simulations. See Section 3.5.1 "Rejection" on page 34.

Too many lines in the file with true parameters '. . . ', more than observed data set!
If a file with the true model parameters is passed it has to contain as many lines as observed data sets are passed to `ABCestimator`. Compare the number of lines of this file with the number of lines in the file containing the observed data sets passed with the parameter `obsName`. See Section 3.9 "Quantile of the true Parameter values within the Posterior Distribution" on page 35.

Wrong number of arguments!
`ABCestimator` was launched with the wrong number of arguments, most likely without the name of a valid `.input`–file. See Section 3.1 "Launching `ABCestimator`" on page 32.

4 Additional Tools

ABCtoolbox includes a set of small programs or scripts that can be useful when performing some ABC estimates. Here we provide a complete list of all additional tools and give some explanations on how to use them.

4.1 Using transformer

This small command line program is designed to perform the linear transformations of summary statistics, just as ABCsampler does them on a file containing the original summary statistics. transformer needs a file with the definitions of the linear transformation as its first argument. This file has exactly the same specifications as the one for ABCsampler (see Section 2.8 "Linear transformed statistics" on page 20). The second argument is the file with the original summary statistics. This file must contain a header line, where the names of the summary statistics correspond to those in the file with the definitions of the linear transformations. This file may also contain additional columns such as model parameters. In this case transformer does the following: all columns prior to the first summary statistics column are printed without any change. Columns after the first summary statistics column, but with a name not listed in the file with the definitions of the linear transformations, will be ignored. This offers the possibility to transform an output file of ABCsampler containing first the model parameters and then the summary statistics at once and even to choose linear combinations of a subset of the summary statistics calculated. Note that on a Unix environment there is also a simple way to remove some columns from a file in general:

```
$ cut -f1-4,6-10 old.txt > new.txt
```

for instance, copies the columns 1 to 4 and 6 to 10 form the file "old.txt" to a new file named "new.txt" in this example. This is a simple way to remove, for instance, some extra model parameter columns. The last argument of transformer is the desired filename of the transformed file. All together, on a Unix environment, simply call transformer like this:

```
$ transformer def.txt sims.txt new_sims.txt
```

where "def.txt" is the file with the definitions of the linear combinations, "sims.txt" the file with the original summary statistics and "new_sims.txt" the desired filename of the output file. Just as ABCsampler, transformer offers the possibility of applying a Box–Cox transformation to the statistics prior to the computation of the linear combinations. To achieve this, simply call transformer with the additional tag "boxcox":

```
$ transformer def.txt sims.txt new_sims.txt boxcox
```

Be careful that the file "def.txt" has the right format (see Section 2.8 "Linear transformed statistics" on page 20).

4.2 Using cumuldens

This small command line program calculates the positions of some locations of interest within the cumulative density distribution of a sample. This may be useful to check for a potential posterior bias (see Section 5.3.2 "Checking for biased posteriors" on page 48). cumuldens requires two obligatory arguments: a file with samples from which the density distribution has to be estimated and a file containing the locations at which the density of the cumulative distribution is of interest. Both files require a header line with the names of the different columns. cumuldens will output the density of the cumulative density distribution of all columns which are present in both files. If a column is missing in the file with the samples, cumuldens will throw an error. On the other hand, additional columns may be present in the file with the samples. This is especially handy, since the output of ABCsampler contains not only the parameters, but also the summary statistics. To check for a potential posterior bias (see Section 5.3.2 "Checking for biased posteriors" on page 48), however, only the model parameter columns are important. On a Unix environment, simply call cumuldens like this:

```
$ cumuldens samples.txt locations.txt
```

where "samples.txt" is the file with the samples from the distribution (e.g. output file of ABCsampler) and "locations.txt" the name of the file containing the locations of interest (e.g. the true model parameter values). Different to transformer, cumuldens writes the results directly to the standard output. `cumuldens` can be told to print the names of the columns used as a first row. Simply call `cumuldens` with the additional tag "`header`":

```
$ cumuldens samples.txt locations.txt header
```

4.3 Using `strStats`

`strStats` is a small command line program that calculates some very basic summary statistics when using micro satellites (STRs). It requires a valid `Arlequin 3.1` input file to read the sampled micro satellites. The outputted summary statistics are the variation in repeat length per population and the differences in the average repeat length between all pairs of populations. Be careful to transform the micro satellites alleles into repeat length when using `strStats`. `simcoal2` writes `Arlequin 3.1` files with repeat length automatically. In total, `strStats` requires 5 arguments:

1st The name of an `Arlequin 3.1` file.
2nd The name of the output file to which `strStats` writes the calculated summary statistics.
3rd An indication whether (1) or not (0) to write a header line with the names of the summary statistics.
4th An indication whether to append to (1) or to overwrite (0) the output file.
5th An indication whether to output the summary statistics for each locus individually (1) or the mean over all loci (0).

A valid program call looks therefore somewhat like this:

```
$ strStats humans.arp sumstats.txt 1 1 0
```

Note that the arguments are almost identical to those required by `arlsumstat`. Be also aware that `strStats` may produce unpredictable results if used with DNA sequences or a wrong data format. Please check the manual of `Arlequin 3.1` at *www.cmpg.unibe.ch* for a definition of the data format.

4.4 Using `glm`

R-script—————————- —

`glm` is a small command line program generating summary statistics under a general linear model GLM. It is intended to be used for testing purposes since it is very fast in generating summary statistics using a GLM and second, the degree of complexity can be arbitrarily chosen. Basically, `glm` generates summary statistics under the GLM

$$\mathbf{s}|\boldsymbol{\theta} = \mathbf{C}\boldsymbol{\theta} + \mathbf{c}_0 + \boldsymbol{\epsilon}, \qquad (8)$$

where $\mathbf{s}$ is an n–dimensional column vector of summary statistics, $\boldsymbol{\theta}$is an m–dimensional column vector of parameter values, $\mathbf{C}$ is an $n \times m$-matrix of constants, $\mathbf{c}_0$ a $n \times 1$-vector and $\boldsymbol{\epsilon}$ a random vector with a multivariate normal distribution of zero mean and covariance matrix $\boldsymbol{\Sigma}_s$:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_s). \qquad (9)$$

The covariance matrix $\boldsymbol{\Sigma}_s$ may encapsulate correlations between the summary statistics, as are normally observed in many population genetics models and likely in others as well. `glm` takes the following arguments:

1st The name of a file with the definitions of $\mathbf{C}$, $\mathbf{c}_0$ and $\boldsymbol{\Sigma}_s$ (see below).
2nd The name of the output file to which `glm` writes the calculated summary statistics.
3rd The value of the first parameter component θ_1.
... The values of the remaining parameter components θ_k, $k = (2, \ldots, m)$.

A valid program call may look as follows:

```
$ glm matrices.txt sumstats.txt 0.34 0.984 1.35
```

The file containing the definitions of $\mathbf{C}$, $\mathbf{c}_0$ and $\boldsymbol{\Sigma}_s$ has the following format: each of the three matrices begins with the tag "C", "c0" and "Sigma", respectively, followed by lines containing the rows of these matrices. Note that $\mathbf{c}_0$, despite being a column vector, has to be coded as a row vector. An example of such a file is given in Figure 8. Note that $\boldsymbol{\Sigma}_s$ must be a symmetric matrix. If no correlations between the summary statistics are assumed, the $\boldsymbol{\Sigma}_s$ is a diagonal matrix containing only the variances of the different summary statistics. The output file of `glm` contains a header row with the names of the statistics ("Stat_1", "Stat_2", ..., "Stat_n") and a second line with the values of the different statistics. `glm` can easily be used with `ABCsampler`: simply use `glm` as the simulation program and do not use any program to calculate summary statistics. The arguments to

```
C
0.2565644169  0.3998737261  0.6191035043   0.0881405491  0.5812833798
0.8341390323  0.8417301406  0.5533106020   0.4275868451  0.3839495266
0.1004940239  0.1214807089  0.9599180175   0.5112989354  0.6948176981
0.3804506403  0.7428206834  0.9395745955   0.2937300280  0.7716452195
0.8394088270  0.4887209875  0.5011698930   0.1772321309  0.6622808776
c0
1  1  1  1  1  1
Sigma
1  0  0  0  0
0  1  0  0  0
0  0  1  0  0
0  0  0  1  0
0  0  0  0  1
```

Figure 8. A Matrix definition file for `glm` *This is an example of a file containing the definitions of the matrices used by the program `glm`. Note that it may not contain any comments. See text for explanations.*

pass to `glm` can be specified with the parameter `simParam` of `ABCsampler`. The parameters defined in the `.est`–file can be used as arguments directly (see Section 2.4 “Using different Simulation Programs” on page 13). The name of the output file of `glm` has to be specified to `ABCsampler` with the parameter `simDataName`. Since `glm` is a very quick way to generate summary statistics from an arbitrary complex model, it is very well suited to test several runs of `ABCsampler` under various conditions. For instance, the convergence of MCMC without likelihoods or Population Monte Carlo may be tested that way. Note that `ABCsampler` offers an internal version of this program which is much faster to run, since the file with the matrix definitions is only read once. To use this internal version, simply set the parameter `simulationProgram` to the tag “INTERNALGLM”, but use the arguments to pass as described above (see Section 2.4.2 “INTERNALGLM” on page 16).

4.5 Plotting Posteriors

`ABCtoolbox` features also an R script, named `plotPosteriors.r`, to plot posteriors from the output of `ABCestimator`. This script reads an `.input`–file of `ABCestimator` and produces pdf plots for each data set for which estimates exist. It needs the original simulation file (the file containing all simulations), the file with the observed statistics and the output files of `ABCestimator` containing the posteriors estimates to be located in the same directory as the `ABCestimator .input`–file. To launch this script on the command line on a Unix platform simply type

```
$ R --vanilla inputfile < plotPosteriors.r
```

where `inputfile` corresponds to the name of the `ABCestimator .input`–file to be used. The default is a file names “abc_glm.input” located in the same directory as the script. To use the script under Windows, simply copy it into the same directory as the `ABCestimator .input`–file and change the default name in the script.

The provided R script plots the prior distribution (estimated from the first 50,000 simulations in the simulation file) in black and the obtained posterior distributions in red. If the file with the retained simulations is written by `ABCestimator` (requested with the parameter `writeRetained`), an additional blue line with the marginal parameter distribution among the retained simulations is plotted.

4.6 Defining PLS components

if the dimensionality of the summary statistics space is large, a reduction using linear combinations may be advantageous. Wegmann and Excoffier (2008) proposed to use PLS components. The definition and use of PLS components is described in detail in Section 5.2 “Choosing Summary Statistics” on page 44, as is the use of `findPLS.r`, a simple R script distributed with `ABCtoolbox` and given in Figure 9.

5 Pitfalls, Hints, and Suggestions

5.1 Defining Priors

As described in Section 6.1 “Bayesian statistics” on page 54, prior distributions are essential when performing Bayesian inference. They do and should have an influence on the outcome of the inference and are recommended to be chosen wisely. In general, the prior distribution are summarizing the *a priori* knowledge on the model parameters in question. Their choice may, however, also have a large impact on the performance of the ABC algorithm. While standard MCMC approaches accept new steps based on likelihood rations, ABC algorithms accept parameter vectors based on the absolute likelihood. This has some severe impact on the acceptance rate which is generally much lower for ABC approaches. Choosing complex models, too many summary statistics or very wide priors may reduce the absolute likelihood and in consequence the acceptance rate of ABC algorithms.

When prior distributions are covering several orders of magnitude, it might be advisable to choose them on a logarithmic scale or otherwise the sampling effort of ABC approaches may concentrate on large values. In such cases the main interest is usually anyway in the order of magnitude of the parameter. In most such cases also the summary statistics depend on the parameter rather on a logarithmic scale than a natural one. It might therefore be wise to parameterize the model with a parameter on the logarithmic scale and to estimate the parameter also on this scale, especially when using a post–sampling adjustment.

5.2 Choosing Summary Statistics

An important approximation is the replacement of the full data by a set of summary statistics (see Section 1.1.3 “Summary Statistics” on page 5). The hope is to capture the whole information contained in the data about the model parameters, while being able to compute a distance between two data sets. That is, we aim at finding a set of sufficient summary statistics for the model in question. While increasing the number of summary statistics may indeed increase the amount of information available for an ABC algorithm, other issues may arise. Basically, the larger the number of summary statistics, the larger the statistical noise included in the posterior estimation (Joyce and Marjoram 2008). The reason is twofold: firstly, many statistics may carry only very limited amount of information about the model parameters and can therefore be considered as just adding a random number. Secondly, in a large summary statistics space, it becomes very difficult to obtain simulations close to the observation and all simulations will have very similar distances to the observed data set, a phenomenon known as “the curse of dimensionality”. Which in turn makes it very difficult to choose the right simulations in the rejection step. Wegmann and Excoffier (2008) indeed showed that when too many summary statistics are included, the obtained posteriors may be biased.

Joyce and Marjoram (2008) proposed to solve this trade-off between the information added by additional summary statistics and the corresponding additional stochastic noise by scoring the different summary statistics based on their impact on the inference. That way they find a minimal set of summary statistics having the largest impact on the inference, which is thought to be an optimal set of summary statistics. Wegmann and Excoffier (2008) proposed a different solution to the problem by aiming at defining a set of orthogonal linear-combinations of summary statistics explaining best the variance in the model parameter space. This is achieved by transforming the summary statistics via Partial Least Squares PLS (Boulesteix and Strimmer 2007). Like principal component analysis (PCA), PLS extracts orthogonal components from a high dimensional data set X of predictor variables, but in addition, these components are chosen such as to appropriately explain the variability of a matrix of response variables by maximizing the covariance matrix of predictor and response variables (see e.g. Tenehaus et al. 1995). As noted by Wegmann and Excoffier (2008) it might be desirable to linearize the relationship between model parameters and statistics before transforming them linearly in some cases. They propose to use a Box–Cox transformation to the statistics:

$$s_{BoxCox} = \frac{s^{\lambda} - 1}{\lambda(\mathrm{GM}(s))^{\lambda - 1}}, \tag{10}$$

```
library("pls"); library("MASS");
#read file with simulations
numComp<-20;
directory<-'/home/me/ABC'; filename<-'someSims.txt';
a<-read.table(paste(directory, filename, sep=''), header=T, nrows=10000, skip=0);
print(names(a));
stat<-a[,16:61]; param<-a[,1:15]; rm(a);
#standardize the params
for(i in 1:length(param)){param[,i]<-(param[,i]-mean(param[,i]))/sd(param[,i]);}
#force stat in [1,2]
myMax<-c(); myMin<-c(); lambda<-c(); myGM<-c();
for(i in 1:length(stat)){
    myMax<-c(myMax, max(stat[,i])); myMin<-c(myMin, min(stat[,i]));
    stat[,i]<-1+(stat[,i] -myMin[i])/(myMax[i]-myMin[ i]);
}
#transform statistics via boxcox
for(i in 1:length(stat)){
    d<-cbind(stat[,i], param);
    mylm<-lm(as.formula(d), data=d);
    myboxcox<-boxcox(mylm, lambda=seq(-20,100,1/10), interp=T, eps=1/50);
    lambda<-c(lambda, myboxcox$x[myboxcox$y==max(myboxcox$y)]);
    myGM<-c(myGM, mean(exp(log(stat[,i]))));
}
#standardize the BC-stat
myBCMeans<-c(); myBCSDs<-c();
for(i in 1:length(stat)){
    stat[,i]<-(stat[,i] ^lambda[i] - 1)/(lambda[i]*myGM[i] ^(lambda[i]-1));
    myBCSDs<-c(myBCSDs, sd(stat[,i]));
    myBCMeans<-c(myBCMeans, mean(stat[,i]));
    stat[,i]<-(stat[,i] -myBCMeans[i])/myBCSDs[i];
}
#perform pls
myPlsr<-plsr(as.matrix(param)   as.matrix(stat), scale=F, ncomp=numComp,
validation='LOO');
#write pls to a file
myPlsrDataFrame<-data.frame(comp1=myPlsr$loadings[,1]);
for(i in 2:numComp){myPlsrDataFrame<-cbind(myPlsrDataFrame, myPlsr$loadings[,i]);}
write.table(cbind(names(stats), myMax, myMin, lambda, myGM, myBCMeans, myBCSDs,
myPlsrDataFrame), file=paste(directory, 'PLSfile_', filename, sep=''), col.names=F,
row.names=F, sep='', quote=F);
#make RMSEP plot
pdf(paste(directory, 'RMSEP_', filename, '.pdf', sep=''));
plot(RMSEP(myPlsr));
dev.off();
```

Figure 9. An R script to find PLS components *This is an example of an R–script to find appropriate PLS components. The scripts writes the definition of the components to a file which has the appropriate format for* `ABCsampler`*. It also writes the RMSEP plots to a pdf (see text). Note that an RMSEP plot can also be generated if no validation is done (removing validation='LOO' from the plsr call). In this case the prediction error is approximated, which is much faster.*

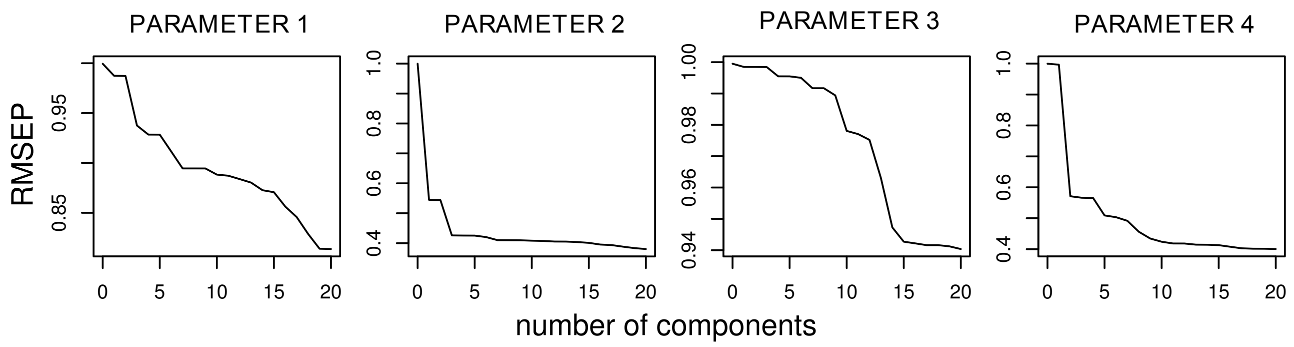


Figure 10. RMEP plot *The script **findPLS.r** writes a file containing RMSEP (root mean squared error) plots for all model parameters (see text). Based on these plots the number of PLS components to use is defined. The figure shows four RMSE plots from a real situation. As can bee seen, the summary statistics do not provide the sam eamount of information for the different parameters. For parameter 3, for instance, even the inclusion of 20 PLS components reduces the predication error by only 6%. The first four PLS components, on the other hand, seem to contain all the information contained in the summary statistics about parameter 2. In this situation it is recommended to work with 5 to 7 PLS components.*

where λ is the power parameter and GM(s) the geometric mean of the statistics. `ABCsampler` offers the possibility to use linear transformed statistics with or without Box–Cox transformation.

5.2.1 Defining PLS Components

`findPLS.r`, a simple R script to define PLS components, is distributed with `ABCtoolbox` and given in Figure 9. It uses the freely available library "PLS" (Mevik and Wehrens 2007). In order to use it for your pourposes, simply change the variables `directory` and `filename` to match a file containing several thousand simulations in the format of an `ABCsampler` output file. Specify the appropriate columns as parameters and statistics in line 7. You may exclude some parameters and statistics at this stage, but we recommend to start with the full set. Be aware that `numComp` has to be smaller or equal to the number of summary statistics.

`findPLS.r` will write two output files: the first is a file that can be used to transform the summary statistics (see below). The second one is a pdf used to define the number of PLS components to use. It contains plots of the prediction error (root mean squared error, RMSEP) when using a given number of PLS components relative to the prediction error without any PLS component. This plot allows to decide on the number of PLS components to use: the optimal set of PLS components is the smallest set carrying a large amount of information about the model parameters. Don't use too many components (around 10 PLS components are generally fine). Examples of RMSEP plots and some interpretations are given in Figure 10 It is also recommended to check if the estimates depend heavily on the number of components used. The RMSEP plot gives also information about the relationship between model parameters and statistics: if any number of PLS components reduces the prediction error of a given model parameter only very little, there is not much hope in estimating this model parameter precisely.

5.2.2 Using Linear Transformed Statistics

In the case of a standard sampling run the easiest way to use linearly transformed summary statistics is to use some of the simulations of the final set and then to transform the large file with all simulations using `transformer` (see Section 4.1 "Using `transformer`" on page 41). In the case of an MCMC without likelihoods or a PMC run this is impossible, since the linear combinations should already be used during the run. Wegmann and Excoffier (2008) proposed to use the calibration simulations of an MCMC without likelihoods run to define PLS components. This is a clever way since these simulations are indeed drawn randomly from the prior and do not require the computation of a distance. Basically, it is advisable to perform a standard sampling run with `ABCsampler` to gen-

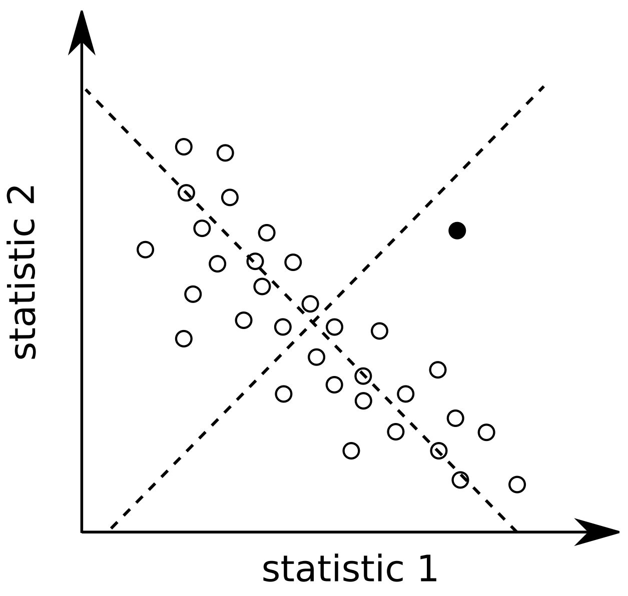


Figure 11. Wrong model *Here we show the difficulty to realize if it is impossible to generate the observed data under a given model. The empty dots represent simulations performed under the model, the black dot represents the observed data* $\mathbf{s}_{obs}$*. While* $\mathbf{s}_{obs}$ *falls well within the marginal distributions of the two summary statistics, it is far away from them in a multivariate space. Sometimes it is beneficial to rotate the coordinate system in order to see this problem. For instance, by performing a principal component analysis (PCA, dashed lines), the discrepancy between* $\mathbf{s}_{obs}$ *and the simulated data becomes visible in the marginals.*

erate the calibration simulations and to define the linear combinations based on these simulations, and finally to pass the calibration simulations as a file to `ABCsampler` for the MCMC without likelihoods run, together with the file containing the definitions of the linear combinations (see Section 2.11.1 "Calibration File" on page 23). Note that the same procedure may also be used when performing a PMC run (see Section 2.11.2 "Launching a Population Monte Carlo algorithm" on page 23).

5.3 Validation

A main difficulty of an ABC analysis is to gain confidence in the obtained results. As was shown by Leuenberger and Wegmann (2009) and others, ABC does indeed lead to an approximation of the true posterior distribution and the assumptions under which it does so, are known. But ABC always leads to posterior distributions, whether the assumptions underlaying it are met or not. It is therefore an absolute necessity to perform some additional analysis to gain confidence in the results obtained. Here we discuss several of these issues and propose ways to validate the results.

5.3.1 Using a Wrong Model

"All models are wrong, but some are usefull." A famous yet true quote from George Box. It is important to note that all Bayesian estimates require an underlying model and this model is at best a good approximation of the truth. This has an important implication: the obtained posteriors reflect the probability of the model parameters given the observed data and the assumed model. If the assumed model is very far from the truth the estimates are meaningless. A problem of ABC is that a likelihood of zero does not exist. Imagine an observed data set and model with which it is impossible to generate the observed data. The likelihood of the observed data under this model is therefore zero. However, it is always possible to select simulated data sets that are closer to the observed data set than others. And it is also possible to perform a post sampling regression adjustment such as ABC-GLM in this case. The danger is, however, that the linear relationship between the model parameters $\boldsymbol{\theta}$ and the statistics $\mathbf{s}$ is estimated in a region that does not encompass the observed data set. To obtain an approximate likelihood at $\mathbf{s}_{obs}$ requires an extrapolation of this relationship, which may give very wrong results. Note that the ABC-REG approach suffers from a similar problem.

To see if an extrapolation happens, it is a good idea to plot the density distribution of the statistics of the retained simulations and to see if the observed statistics fall well within this range. A problem arising in higher dimensions of summary statistics is that the observed data may fall within the marginal distribution of the retained summary statistics, but not in the multivariate distribution. In Figure 11 this is shown in two dimensions. While it is impossible to plot a high dimensional space, it might be useful to plot all combinations of parameters to check if the observed data falls within the obtained clouds. Performing a coordinate rotation, such as a principal component analysis (PCA), is sometime helpful to see a discrepancy between the observed and simulated data even in the marginals (see Cornuet et al. 2008 and Figure 11). `ABCestimator` reports a p-value for the observed data set under the estimated GLM, which can also be used to judge, if the observed data is in agreement with the simulated data (see Section 3.10 "Output" on page 36). A way to check if the observed data is plausible under the estimated model, is to simulate data sets under the estimated model (for instance by taking the mode of all parameter estimates) and to check if the observed summary statistics can be reproduced.

If the observed data is not reproduceable with the model used, consider changing the model. Be aware that the priors on the model parameters are part of the model and do have an influence on the posterior estimates, as they should. So choosing the priors wisely is an important aspect of running an ABC estimation.

5.3.2 Checking for biased posteriors

Since there is no general setting for any ABC setting, it is advisable to test if the obtained posteriors of an ABC estimation are biased or not. There is a simple property of any real posterior distribution, obtained with ABC or not: the posterior distribution $\pi(\boldsymbol{\theta}|\mathbf{s})$ denotes the probability of a given model parameter vector $\boldsymbol{\theta}$ to have produced the data set $\mathbf{s}$ under the assumed model $f(\mathbf{s}|\boldsymbol{\theta})$ and under the assumed prior distribution. This property can easily be checked by generating a set of pseudo–observed data sets under the model with parameters drawn from the posterior distributions. The positions of the true parameters within the posterior distributions have to reflect the probability denoted by the posterior distributions. Basically, their position in the marginal cumulative posterior distribution has to be uniformly distributed, as we proof in the following (see also Cook et al. 2006):

Let $f(\mathbf{s}|\boldsymbol{\theta})$ be a continuous model with p-dimensional parameter vectors $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$ (defined on some domain $\mathbf{\Pi} \subseteq \mathbb{R}^p$) yielding n-dimensional summary statistics $\mathbf{s} \in \mathbb{R}^n$. For a prior distribution $\pi(\boldsymbol{\theta})$ we denote by $\pi(\boldsymbol{\theta}|\mathbf{s})$ the posterior and by

$$\pi_k(\theta_k|\mathbf{s}) = \int \pi(\boldsymbol{\theta}|\mathbf{s}) d\boldsymbol{\theta}_{-k} \tag{11}$$

the marginal posterior of the k-th parameter component θ_k (integration is over all parameter components but θ_k).

Theorem. *If a parameter $\boldsymbol{\theta}$ is drawn $\sim \pi(\boldsymbol{\theta})$ and $\mathbf{s}$ is a summary statistics produced by the model $f(\mathbf{s}|\boldsymbol{\theta})$ for the chosen parameter then the random values*

$$a_k = \int_{-\infty}^{\theta_k} \pi_k(\theta_k|\mathbf{s}) d\theta_k, \quad k = 1, \ldots, p, \tag{12}$$

are uniformly (but not in general independently) distributed on the interval $[0, 1]$.

Proof. The joint distribution of $\boldsymbol{\theta}$ and $\mathbf{s}$ is $f(\mathbf{s}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$. Given some fixed value $\mathbf{s} = s \in \mathbb{R}^n$ for which $\int f(s|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta} > 0$, the conditional density of θ_k is

$$\pi_k(\theta_k|\mathbf{s} = s) = \frac{\int f(s|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}_{-k}}{\int f(s|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}}. \tag{13}$$

Hence

$$a_k(\theta_k|\mathbf{s} = s) = \int_{-\infty}^{\theta_k} \pi_k(\theta_k|\mathbf{s} = s) d\theta_k \tag{14}$$

has a uniform distribution as follows from the well–known fact that $F(\mathbf{X})$ is uniformly distributed on $[0, 1]$, whenever $\mathbf{X}$ is an absolutely continuous random variable with cumulative distribution function $F(\cdot)$. Since the distribution of $a_k(\theta_k|\mathbf{s} = s)$ does in fact not depend on $\mathbf{s}$, the claim follows. □

`ABCestimator` outputs the positions in the marginal cumulative posterior distribution a_k

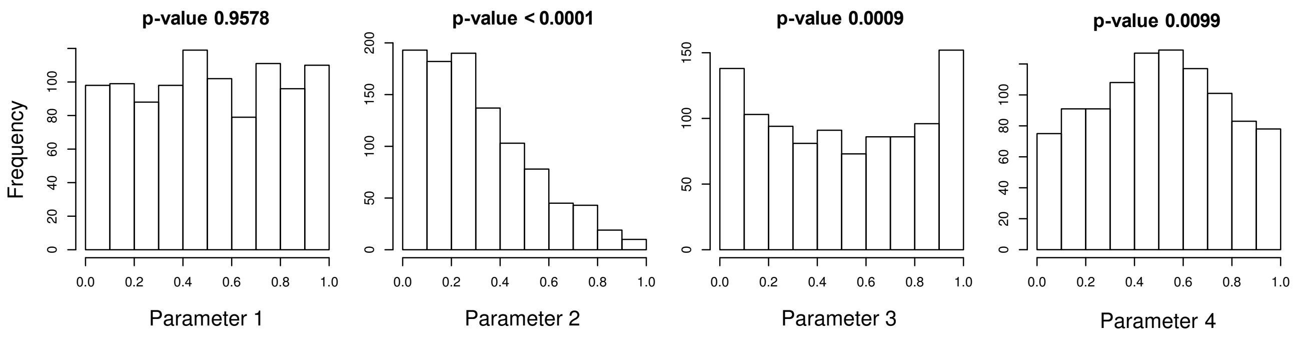


Figure 12. Posterior Quantiles *The figure shows the distribution of the posterior quantiles for a model with four parameters. The p–values reported above the histograms was computed with a Kolmogorov–Smirnov test against an uniform distribution. While parameter 1 seems to be unbiased, the other three parameters show some severe departure from uniformity. Parameter 2 is biased toowards too large values. The posterior distributions of parameter 2 are too narrow as too many times the true values lie within the tails (or even outside) of the distribution. Estimates of parameter 4, on the other hand, are too broad with too many true values falling in the middle of the posterior distribution.*

(posterior quantiles) directly if the true model parameter values are provided (see Section 3.10 "Output" on page 36). If no post–sampling regression adjustment is performed, the program `cumuldens` may be used (see Section 4.2 "Using `cumuldens`" on page 41). To test for uniformity we recommend to perform a Kolmogorov–Smirnov test against a uniform distribution. The corresponding command in R is `ks.test(q, ''punif'')'`, where `q` is a vector with the posterior quantiles.

Interpreting the Distribution of Posterior Quantiles In Figure 12 we report the distribution of the posterior quantiles of a model with four parameters. For each parameter the p–value computed using a Kolmogorov–Smirnov test is reported. While the first parameter is most likely unbiased, the other three parameter show departure from uniformity. Parameter 2 is an example of a biased parameter, since the true value was found too many times in the left half of the posterior distribution. This suggests that this parameter is usually overestimated. Parameter 3 and 4 exhibited often too narrow and too wide posteriors, respectively. The true value of parameter 3 was found too often in the tails of the posterior distribution. The true value of parameter 4, on the other hand, was found too many times in the center of the distribution.

Once posterior biases have been found, there are two ways of dealing with them: If the biased parameter is not of any interest itself (e.g. a nuisance parameter), it is possible to ignore its biased. But an interpretation of biased parameters should be avoided. In many case it might however be better to try to change the model in order to get unbiased results. Here we propose three potential modifications:

- Reduce the complexity of the model
- Shrink prior ranges
- Reduce summary statistics space

To get unbiased estimates in complex models is very difficult. Note that a model with more than five parameters can already be very complex. We recommend to start with a really simple model and to complexify if estimates are feasible. A way to reduce the complexity of a model is also to reduce the prior ranges. Keep in mind that parameters ranging over several orders of magnitude are recommended to be estimated on a logarithmic scale. Some hints on how to choose prior distributions are given in Section 5.1 "Defining Priors" on page 44. Some times a reduction of the summary statistics space might do the job. Actually, Wegmann and Excoffier (2008) have shown that in population divergence model with only four parameters, the inclusion of too many summary statistics creates biased posteriors. How to choose summary statistics wisely is discussed in Section 5.2 "Choosing Summary Statistics" on page 44.

5.4 Using ABCtoolbox with simcoal2 and arlsumstat

While the software package ABCtoolbox can be used with almost any simulation program or program to compute summary statistics, we used the two programs simcoal2 and arlsumstat to demonstrate the usage of ABCsampler. Both programs are freely available at *www.cmpg.unibe.ch* and shiped with this package. In this section we give some instructions on how to use ABCsampler with these two programs. Note that the distribution of ABCtoolbox contains the example files used here.

The first step is to set up simcoal2 to simulate genetic data under the desired model (see also *www.cmpg.unibe.ch/software/* for details). We will use here the same example used in the Section 2 "Using ABCsampler" on page 11 and depicted in Figure 2. Consider a population of size *N_NOW* that was larger in the past and changed its size *T_SHRINK* generations ago. The ancestral population size is given by *N_ANCESTRAL*. simcoal2 reads the parameters of the model from a file. The manual of simcoal2 contains detailed explanations on how to set up different population genetic models (see *www.cmpg.unibe.ch/software*). The file for the simple example considered here is given in Figure 5. Note that instead of values, the input–file for simcoal2 contains special tags at specific locations. This is because we will tell ABCsampler to parse this file and thereby to replace these tags with the values of the model parameters drawn from their prior distributions. To achieve this, we first need to define which tags correspond to model parameters and to define their prior distributions. This is done within a specific file termed .est–file. A possible .est–file for the simple model used here is given in Figure 4. See Section 2.3 "Defining prior distributions - .est File" on page 11 for a detailed description on how to set up an .est–file. The name of the .est–file used is told ABCsampler with the parameter estName in the .input–file of ABCsampler (see Section 2.3 "Defining prior distributions - .est File" on page 11). The name of the input–file for simcoal2 is given by the parameter simInputName (see Section 2.4 "Using different Simulation Programs" on page 13).

We also need to explain ABCsampler how to use the simulation program, simcoal2 in this case. This is done with the two parameters simulationProgram and simParam (see Section 2.4 "Using different Simulation Programs" on page 13). The first of them specifies the name of the executable to use (simcoal2 in this case), the second one the arguments that have to be passed to the executable. simcoal2 takes four arguments: the name of its input–file, the number of simulations to perform, whether genotypic data is simulated (0/1) and wether the gametic phase is known (0/1) . Of course, simcoal2 is requested to perform a single simulation per launch. Lets assume we aim at genotypic data with unknown gametic phase. For simcoal2 the parameter simParam may therefore be specified like this: simParam SIMINPUTNAME#1#1#0, where SIMINPUTNAME is a special tag telling ABCsampler to pass the name of the parsed input file name for simcoal2, specified by simInputName (see Section 2.4 "Using different Simulation Programs" on page 13). Note that the parsed file will have a "-temp" added before the extension. using the tag SIMINPUTNAME is therefore very convenient, since ABCsampler handles this issue.

We then need to tell ABCsampler to use arlsumstat to compute summary statistics from the output of simcoal2. This is done with the parameter sumStatProgram. We also have to tell ABCsampler which arguments have to be passed to arlsumstat. This is done with the parameter sumStatParam. arlsumstat takes four arguments: the name of a valid Arlequin file, the name of the file to which arlsumstat will write the summary statistics, wether to append (1) or overwrite the output file and wether to write a header line with the names of the summary statistics (0/1). simcoal2 writes directly a valid Arlequin file with a name that is a derivate from the input file used: simcoal2 replaces the extension with "_0.arp". The name is therefore a construction using the simulation program input filename, specified to ABCsampler with the parameter simInputName (see above). Note that we can use SIMINPUTNAME in this case as an identifier for the part of the filename identical to the name of the input file of the simulation program (without extension).

In order to compute the desired summary statistics, arlsumstat requires two files to be present in the same directory: arl_run.ars and ssdefs.txt. The first file is a configuration file telling arlsumstat which summary statistics to compute. arlsumstat is basically a command line version of Arlequin ans uses the same configuration file (see the Arlequin manual at *www.cmpg.unibe.ch* for explanations). Since the structure of this file is not very obvious, it is a good practice to use the graphical version of Arlequin to produce it. The second file (ssdefs.txt) is a file to tell arlsumstat which summary statistics to print. It must contain two header lines. The next two lines contain two tags: POP_LEVEL and GROUP_LEVEL. They indicate on which level the summary statistics will be computed. To switch off a level, simply add a "#" sign in front of the tag. If arlsumstat is requested to print the statistics on the population level, the same summary statistics are printed for each population. Similarly, the same summary statistics will be printed for each group, if requested. The numbering follows the order in the input–file. Note that the groups are defined in the input–file, which has to be a valid Arlequin–file (see the Arlequin manual at *www.cmpg.unibe.ch* for explanations). The next lines of the ssdef file contain the names of the summary statistics to be printed. Again, putting a"#" in front of a tag will stop a given statistics to be printed. For each statistics several tags are available. The prefix ALL will print the given statistics for each population or group. The prefix ALLSD will print the standard deviation of this statistics over loci for each population. The next prefixes tell arlsumstat to print the mean (MEAN) or the standard deviation (SD) of the summary statistics over populations / groups. The prefix TOT finally will print the summary statistics computed as if all samples were pooled to a single population. Were applicable, the prefix PAIRWISE will print the summary statistics computed for all pairs of populations. An example file, which can be modified easily, is found in the subfolder exampleFiles.

The file, to which arlsumstat writes the computed summary statistics, has also to be known to ABCsampler and has to be specified with the parameter sumStatFile (the default value is summary_stats_temp.txt, see Section 2.5.1 "File with computed summary statistics" on page 17). We may simply use the tag SSFILENAME in the argument list of arlsumstat. It is important to understand that the output file of arlsumstat has to be overwritten each time and has to contain a header line (see Section 2.5.1 "File with computed summary statistics" on page 17). To sum up, the parameter sumStatParam has therefore be specified like this: simParam SIMINPUTNAME_0.arp#SSFILENAME#0#1.

The complete .input file for ABCsampler, containing all the above parameters, is given in Figure 3 and included in the distribution of ABCtoolbox among the example files.

5.5 Parallelizing ABCsampler

Most ABC estimations rely on a huge amount of simulations, several hundred thousand or even several millions. Due to the command line interface of ABCsampler it is very easy to use a grid to perform these simulations without having to install any software on the individual nodes. Simply chunk the total amount of simulations to perform into small pieces and send each piece onto an individual node, together with the compiled version of ABCsampler and the necessary .input–file. A simple bash script that can be used to do this on a grid using the "Sun Grid Engine" is given in Figure 13. Such a script can be launched on the master of the grid with the following command:

```
$ qsub -t 1-100:1 ''script.sh''
```

which will launch an array job of 100 individual jobs numbered form 1 to 100. Since the results of any of these individual jobs is copied into a unique folder (basically the "test_run_" followed by the job number, see Figure 13), the results have to be concatenated before passing to ABCestimator. This can be achieved for instance as follows:

```
$ head -n1 testrun_1/*output*.txt > res.txt
 > for fol in testrun_*
 > do tail -n+2 ${fol}/*output >> res.txt
$ done
```

Be aware that the random generator of ABCsampler is initialized from the current time. If several jobs submitted at the same time to a grid, the random generator might be initialized with the same seed. To circumvent this problem use the parameter addToSeed when launching sample and pass a value that depends on the job submitted, such as the task ID. The provided script uses this possibility.

```
#!/bin/bash
#Settings for the Sun Grid Engine
#$-l s_CPU=20:00:00
#$-l scratch=1,scratch_size=500M,scratch_files=1k
#$-A User
#$-N MCMC
#$-p 0
#create a unique folder and store folders
mkdir testrun_${SGE_TASK_ID}
scratchFolder="/scratch/local/${JOB_ID}.${SGE_TASK_ID}.${QUEUE}"
homefolder=pwd
#copy files
cp test.input test.par test.est test.obs $scratchFolder/.
cp ABCsampler1.0 simulationProgram SSProgram $scratchFolder/.
#go on the node and launch ABCsampler1.0
cd $scratchFolder
chmod +x ABCsampler1.0 simulationProgram SSProgram
./ABCsampler1.0 test.input addToSeed=${SGE_TASK_ID}
#copy results back
cp *output*.txt *.log $homefolder/${SGE_TASK_ID}
```

Figure 13. Bash script to submit *`ABCsampler`* *This is a minimal bash script to submit `ABCsampler` on a grid running with the "Sun Grid Engine". Specific setting for the grid have to be adjusted, especially the scratch folder depends on the individual setup.*

5.6 Parallelizing an MCMC without likelihood run

So far we have only discussed on how `ABCsampler` can be used to perform a single MCMC without likelihoods chain. When introducing their recommendations on how to run an MCMC without likelihoods efficiently, Wegmann and Excoffier (2008) used a parallelized version of their algorithm to demonstrate its applicability. Here we outline briefly how to best achieve this. The basic idea of running the parallelized version is to spread the potentially computation intensive task on several CPUs, for instance on a grid. Therefore, the chain is basically chopped into small parts, each of which can run on an individual CPU. The algorithm proposed by Wegmann and Excoffier (2008), however, involves a calibration step which should not be repeated for each part. The best way to run a parallelized MCMC without likelihoods run seems therefore to first use `ABCsampler` to produce a set of simulations under standard sampling. The output of this run can then be used as the calibration simulations by passing the output file to `ABCsampler` using the parameter `calName`. Of course, the same file can be used for every chain that is launched in parallel. Therefore, parallelizing an MCMC without likelihoods chain is very similar to parallelizing a standard sampling approach. Since the start-up has to be repeated for every individual chain and since the chain may have to run sufficiently long to explore the whole parameter space, it is advisable to make the individual chains too small. Note, however, that each of the individual chains is potentially launched from a different starting position, which may actually help in exploring the parameter space.

5.7 Using different Marker Types with `simcoal2`

The simulation program `simcoal2`, which was used by Wegmann and Excoffier (2008) and others, is only capable of performing DNA or STR loci at once. In order to generate both types of markers at once, `simcoal2` has to be called twice in each iteration. This approach was, for instance, chosen by Wegmann et al. (2009). This is easy to do, since `ABCsampler` provides all necessary tools to perform several simulations with the same model parameter values, but different simulation program input files per iteration. Simply specify two files `simInputName` and two files with observed summary statistics with `obsname`. See Section 2.7 “Several simulations per iteration” on page 18 for a detailed description on this topic.

6 ABC – the Methodological Reference

With the advent of ever more powerful computers and the refinement of algorithms like Markov Chain Monte Carlo (MCMC) or Gibbs sampling, Bayesian statistics has become an important tool for scientific inference during the past two decades. While this is true for virtually any field of modern science, the rise of such methods was especially pronounced in the area of population genetics (Marjoram and Tavare 2006). And since the authors are active in this discipline, all examples and models used arise from this field of research. Note, however, that the methods are equally well applicable to any other model or question.

While detailed description of the implemented ABC algorithms are given, we refer the reader to the original papers for a deep understanding of the methodology.

6.1 Bayesian statistics

Consider a model $\mathcal{M}$ creating data $\mathcal{D}$ (DNA sequence data, for example) determined by parameters $\boldsymbol{\theta}$ whose joint prior density we denote by $\pi(\boldsymbol{\theta})$. The quantity of interest is the posterior distribution of the parameters which can be calculated by the Bayesian rule

$$\pi(\boldsymbol{\theta}|\mathcal{D}) = c \cdot f_{\mathcal{M}}(\mathcal{D}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}), \tag{15}$$

where $f(\mathcal{D}|\boldsymbol{\theta})$ is the likelihood of the data and c is a normalizing constant. Consider the following population genetic example: Imagine that the size of a given population is of interest, but unknown. A crude probability distribution of the population size is available from "mark and recapture" data. This distribution is the *a priori* information available and hence the prior for the analysis. Assume that a new effort is done and a collection of individuals from this population are sampled and genetic data becomes available, for instance the number of segregating sites at a given locus. In a Bayesian framework, the first step to do is to set up a model linking the population size, the parameter of interest, with genetic data, in this case the number of segregating sites. Then, the likelihood function is derived for this model. The best available estimate of the population size *a posteriori* of the genetic analysis is then, given by the Bayesian rule in equation 15, proportional to the product of the prior and the likelihood function given the observed data $\mathcal{D}$ (the number of segregating sites in this example). In other words, the probability distribution of the parameters $\boldsymbol{\theta}$ thought to be correct before an experiment was conducted (our *a priori* belief) is changed by the outcome of an experiment (data $\mathcal{D}$).

Direct use of the Bayesian rule (Equation 15), however, is often thwarted by the fact that the likelihood function cannot be calculated analytically for many realistic models. And even if the likelihood function is traceable, it is often impossible to integrate it, which is need to get the constant c. However, if likelihood ratios (the ratio of the likelihood for two different parameter values) can be computed analytically, one possible workaround is to sample from posterior distributions via a MCMC approach. A successfull example is the program IMa (Hey and Nielsen 2007), which estimates demographic parameters of the Isolation with Migration model (Hey and Nielsen 2004) based on MCMC simulations of gene genealogies.

6.2 Likelihood free samplers

In many cases, however, not even likelihood ratios are analytically traceable. In these cases one is obliged to have recourse to stochastic simulation. In fact, the likelihood $f(\mathcal{D}|\theta)$ for any given parameter value θ_i is equal to the frequency of observing $\mathcal{D}' = \mathcal{D}$ among a large sample of simulated data $\mathcal{D}'$ with fixed parameter value θ_i.

6.2.1 Rejection sampling

Tavare et al. (1997) proposed the following rejection sampling method for simulating a posterior random sample directly:

```
REJ1 Draw a parameter vector $\theta_i$ randomly
     from the prior.
REJ2 Simulate $\mathcal{D}'$ using $\theta_i$.
REJ3 Accept $\theta_i$ if $\mathcal{D}' = \mathcal{D}$. Go to 1.
```

The distribution of the accepted θ_i converges exactly to the posterior distribution $\pi(\theta|\mathcal{D})$. Tavare et al. (1997) propose to replace the full data $\mathcal{D}$ by a summary statistics s (like the number of segregating sites in their setting). Even if the statistic is not sufficient for $\mathcal{D}$ – that is, the statistic does not capture the full information contained in the data about the parameters – rejection sampling allows for the simulation of approximate posterior distributions of the parameters in question (the scaled mutation rate in their model). This approach was extended to multiple-parameter models $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$ with multivariate summary statistics $\mathbf{s} = (s_1, \ldots, s_n)$ by Weiss and Haeseler (1998). In their setting, a candidate vector $\boldsymbol{\theta}_i$ of parameters is simulated from a prior distribution and is accepted if its corresponding vector of summary statistics $\mathbf{s}_i$ is sufficiently close to the observed summary statistics $\mathbf{s}_{obs}$ with respect to some metric in the space of $\mathbf{s}$, i.e. if $\text{dist}(\mathbf{s}, \mathbf{s}_{obs}) < \delta_\epsilon$ for a fixed tolerance δ_ϵ.

6.2.2 Summary Statistics

In order to be able to calculate a distance between the simulated data $\mathcal{D}'$ and the observed data $\mathcal{D}$, the data needs to be summarized in a quantitative form. This is achieved by replacing the full data $\mathcal{D}$ by a set of summary statistics $\mathbf{s} = (s_1, \ldots, s_n)$.

Common to all ABC algorithms is their dependence on the absolute likelihood. Since some parts of the data are usually not influenced by the model parameters, replacing the full data by summary statistics may actually increase the absolute likelihood and therefor the acceptance rate of any ABC algorithm. As an example imagine a simple population genetics model of a single population of constant size. A common way to generate DNA sequences for samples from such a population is to first generate a genealogy using the coalescent theory (J. F. C. Kingman 1982; J. Kingman 1982) and second, to sprinkle mutations on that genealogy (Excoffier et al. 2000). This is usually done by assuming a random sequence of DNA (since the true sequence is unknown) at the most recent common ancestor of all samples (the top of the genealogy) and then changing some basepairs at random positions in the genealogy. While the parameter of this simple model (namely $\theta = 2N\mu$, where N is the effective population size and μ the mutation parameter) has a strong influence on the genetic diversity of the sample, the exact basepairs are not influenced. Therefore, by rejecting all simulated data sets where $\mathcal{D}' \neq \mathcal{D}$, we act to stringent. Given a set of summary statistics that are sufficient for $\mathcal{D}$ – that is, the statistics capture the full information contained in the data about the parameters –, a much higher acceptance rate can be achieved. For instance, in our simple model, the number of segregating sites s is sufficient for the parameter θ.

Note, however, that in most cases the set of summary statistics is not sufficient unless a large set of summary statistics $\mathbf{s} = (s_1, \ldots, s_n)$ is used. Assume the summary statistics to be random variables. The probability to have an exact match between $\mathbf{s}$ and $\mathbf{s}_{obs}$ is given by

$$P(\mathbf{s} = \mathbf{s}_{obs})^n, \tag{16}$$

a phenomenon known as “the curse of dimensionality”. Therefore, when the dimension n of summary statistics is high, the tolerance must be chosen rather large or else the acceptance rate becomes prohibitively low. This, however, distorts the precision of the approximation of the posterior distribution with large tolerance ϵ values resulting in posterior distributions dominated by the prior $\pi(\boldsymbol{\theta})$. A way to lower the dimensionality was introduced by Wegmann and Excoffier (2008) and is discussed in the Section 5.2 “Choosing Summary Statistics” on page 44.

6.2.3 MCMC without likelihoods

This situation can be partially alleviated by speeding up the sampling process. Marjoram et al. (2003) developed a variant of the classical Metropolis-Hastings algorithm, which allows them to sample directly from the (approximate) posterior distribution of $\boldsymbol{\theta}$. They showed that a Markov chain where newly simulated data $\mathcal{D}'$ would be accepted if they were equal to the observed data $\mathcal{D}$, and rejected otherwise, would converge to the right posterior distribution. For complex data sets where the acceptance rate is too small, they proposed to replace the full data by summary statistics and to accept new parameter values if sufficiently close to the data, like proposed for the rejection algorithm by Weiss and Haeseler (1998):

MCMC1 If now at $\boldsymbol{\theta}$, propose a move to $\boldsymbol{\theta}'$ according to a transition kernel $q(\boldsymbol{\theta} \to \boldsymbol{\theta}')$.

MCMC2 Simulate $\mathcal{D}'$ using θ_i.

MCMC3 Calculate summary statistics $\mathbf{s}'$ on $\mathcal{D}'$.

MCMC4 If $\text{dist}(\mathbf{s}, \mathbf{s}_{obs}) < \delta_\epsilon$ go to 5, otherwise stay at $\boldsymbol{\theta}$ and go back to 1.

MCMC5 Accept $\boldsymbol{\theta}'$ with probability

$$h(\boldsymbol{\theta} \to \boldsymbol{\theta}') = \min\left(1, \frac{\pi(\boldsymbol{\theta}')q(\boldsymbol{\theta}' \to \boldsymbol{\theta})}{\pi(\boldsymbol{\theta})q(\boldsymbol{\theta} \to \boldsymbol{\theta}')}\right),$$

otherwise remain at $\boldsymbol{\theta}$. Go to 1.

Marjoram et al. (2003) showed that this algorithm converges to $\pi(\theta|\text{dist}(\mathbf{s}, \mathbf{s}_{obs}) < \delta_\epsilon)$, which is a good approximation of $\pi(\theta|\mathcal{D})$ if δ_ϵ is small and $\mathbf{s}$ sufficient for $\boldsymbol{\theta}$.

As discussed earlier (see Section 5.2 "Choosing Summary Statistics" on page 44), the choice of the tolerance distance δ_ϵ is crucial for the performance of the algorithm and depends on the model, the prior definitions and the observed summary statistics $\mathbf{s}_{obs}$. Wegmann and Excoffier (2008) proposed to use an initial calibration set of n simulations (they proposed 10,000), where parameter values are randomly drawn from the prior distribution, such as to obtain an empirical approximation of $\pi(\delta|\mathbf{s}_{obs}, \boldsymbol{\theta})$, which is the expected distribution of distances. With this distribution at hand, a tolerance level ϵ and a tolerance distance δ_ϵ such that $P(\delta \leq \delta_\epsilon) = \epsilon$ can be defined conveniently. For instance, by setting $\epsilon = 0.01$, a tolerance distance δ_ϵ can be defined, such that a simulation with parameters $\boldsymbol{\theta}$ drawn randomly from the prior distribution will have a distance $\delta \leq \delta_\epsilon$ in 1% of the cases. Note also, that among the n simulations generated, all $n\epsilon$ simulations with $\delta \leq \delta_\epsilon$ are usable as starting points for the MCMC chain. Therefore, no burn in is required. Note that the n calibration simulations are also used to get means and standard deviations of all summary statistics. These moments are required to standardize the statistics before the calculation of the Euclidean distance between observed and simulated data sets.

Wegmann and Excoffier (2008) also propose to use the $n\epsilon$ simulations with $\delta \leq \delta_\epsilon$ to adjust the transition kernel. The basic idea is that some parameters have a relatively narrow distribution among the simulations with $\delta \leq \delta_\epsilon$, while others cover a relatively broad range. By adjusting the transition kernel of a given parameter according to the standard deviation of this parameter among the $n\epsilon$ simulations with $\delta \leq \delta_\epsilon$, the parameter space will be explored differently for different parameters.

Since the chain always starts from a position where a simulation resulted in $\delta \leq \delta_\epsilon$ (see above), a burn in is not necessary. Nonetheless, not all starting values are equally likely positions for the chain and, as was shown by Sisson et al. (2007), an MCMC without likelihood chain may stick in regions with small likelihoods because the acceptance rate is proportional to the likelihood. Wegmann and Excoffier (2008) proposed to restart the chain from a new position, again randomly chosen among the $n\epsilon$ positions from the initial calibration step, if the chain does not move within a number of iterations (they used 20 iterations).

Implemented MCMC without likelihood Algorithm To sum up, the algorithm proposed by Wegmann and Excoffier (2008) and implemented in `ABCsampler` goes as follows:

1. Perform n simulations with parameters $\boldsymbol{\theta}'$ randomly drawn from their priors, and each time compute their associated set of summary statistics $\mathbf{S}'$.
2. Compute PLS components from the n $\boldsymbol{\theta}'$ and $\mathbf{S}'$ vectors.
3. For all n simulations, transform their associated summary statistics $\mathbf{S}'$ into k retained PLS components, as $\mathbf{S}'_{PLS}$. Transform the observed summary statistics $\mathbf{S}$ as $\mathbf{S}_{PLS}$ and compute $p_n(\delta|\mathbf{S}_{PLS}, \boldsymbol{\theta})$.
4. Fix ϵ, estimate δ_ϵ from $p_n(\delta|\mathbf{S}_{PLS}, \boldsymbol{\theta})$, and set the proposal range of the parameters (for the transition kernel $q(\boldsymbol{\theta} \to \boldsymbol{\theta}')$ based on φ and the variability of the parameters among the $n\epsilon$ retained simulations.
5. Start an MCMC chain of total length s from a position $\boldsymbol{\theta}$ randomly chosen from the $n\epsilon$ simulations closest to D. Set $i = 0$.
6. If now at $\boldsymbol{\theta}$, propose a move to $\boldsymbol{\theta}'$ according to a transition kernel $q(\boldsymbol{\theta} \to \boldsymbol{\theta}')$. Increment i.
7. Simulate D' based on $\boldsymbol{\theta}'$. Compute the summary statistics $\mathbf{S}'$ and transform them into $\mathbf{S}'_{PLS}$.
8. If $\delta_i = \|\mathbf{S}'_{PLS} - \mathbf{S}_{PLS}\| \geq \delta_\epsilon$ stay at $\boldsymbol{\theta}$ and go to 10.
9. Accept $\boldsymbol{\theta}'$ with probability $\min\left(1, \frac{\pi(\boldsymbol{\theta}')q(\boldsymbol{\theta}' \to \boldsymbol{\theta})}{\pi(\boldsymbol{\theta})q(\boldsymbol{\theta} \to \boldsymbol{\theta}')}\right)$, otherwise stay at $\boldsymbol{\theta}$.
10. If $i < s$ go to 6.

Note that the transformation of the statistics into PLS components is an optional step and other linear transformations may be used.

6.2.4 Population Monte Carlo

Sisson et al. (2007) proposed a sequential Monte Carlo sampler (PRC) claimed to require substantially less iterations than the MCMC without likelihoods. Later, Beaumont et al. (2009) showed that the original formulation leads to biased posteriors and proposed an alternative algorithm termed Population Monte Carlo (PMC) by introducing a weighting scheme based on importance sampling arguments. The main idea is to generate a set of simulations which are closer to $\mathbf{s}_{obs}$ than a threshold distance δ_ϵ that is, $\text{dist}(\mathbf{s}, \mathbf{s}_{obs}) < \delta_\epsilon$. Then, a new set of simulations is performed with model parameters generated from the previous set of parameters according to some transition kernel $q(\boldsymbol{\theta} \rightarrow \boldsymbol{\theta}')$. Beaumont et al. (2009) proposed to use a normal transition kernel. Therefore, the new set of parameter vectors are not drawn randomly from the prior distribution, but from the approximate posterior distribution of the previous iteration $\pi(\boldsymbol{\theta}|\text{dist}(\mathbf{s}, \mathbf{s}_{obs}) < \delta_{\epsilon_t})$. This procedure is repeated with a decreasing series of tolerance values ϵ_t. It is important to realize that a weighting scheme according to the prior distribution is important, since the samples will otherwise converge to the space associated with smallest distances, thereby ignoring the prior distribution (see Beaumont et al. (2009)). The proposed algorithm by Beaumont et al. (2009) is as follows:

```
PMC1 Set $t = 1$.
PMC2 For $i$=1, ..., N, repeat:
        PMC2a Draw a parameters vector $\boldsymbol{\theta}_{i,t}$ randomly from the prior.
        PMC2b Simulate $\mathcal{D}$ using $\boldsymbol{\theta}_{i,t}$.
        PMC2c Calculate summary statistics $\mathbf{s}$ on $\mathcal{D}$.
        PMC2d If $\text{dist}(\mathbf{s}, \mathbf{s}_{obs}) < \delta_t$ accept $\boldsymbol{\theta}_{i,t}$, otherwise go back to PMC2a.
PMC3 Set all weights $\omega_{i,t} = 1/N$.
PMC4 Set $\tau_t^2$ as twice the empirical variance of the accepted $\boldsymbol{\theta}_{i,t}$.
PMC5 Increment $t$.
PMC6 For $i$=1, ..., N, repeat:
        PMC6a Pick a parameters vector $\boldsymbol{\theta}_i^*$ from the accepted $\boldsymbol{\theta}_{i,t-1}$ with probabilities $\omega_{i,t-1}$.
        PMC6b Generate a parameter vector $\boldsymbol{\theta}_{i,t}$ using a normal transition kernel $\boldsymbol{\theta}_{i,t}|\boldsymbol{\theta}_i^* \sim N(\boldsymbol{\theta}_i^*, \tau_{t-1}^2)$.
        PMC6c Simulate $\mathcal{D}$ using $\boldsymbol{\theta}_{i,t}$.
        PMC6d Calculate summary statistics $\mathbf{s}$ on $\mathcal{D}$.
        PMC6e If $\text{dist}(\mathbf{s}, \mathbf{s}_{obs}) < \delta_t$ accept $\boldsymbol{\theta}_{i,t}$, otherwise go back to PMC6a.
PMC7 Set all weights
        $\omega_{i,t} \propto \frac{\pi(\boldsymbol{\theta}_{i,t})}{\sum_{j=1}^N \omega_{j,t-1}\varphi(\frac{1}{\tau_{t-1}^2}(\boldsymbol{\theta}_{i,t} - \boldsymbol{\theta}_{j,t-1}))}$.
PMC8 Set $\tau_t^2$ as twice the weighted empirical variance of the accepted $\boldsymbol{\theta}_{i,t}$.
PMC9 Go back to PMC5.
```

Note that $\varphi()$ denotes the multivariate normal density function. The generated sample at any iteration is a valid sample corresponding to a sample obtained under a rejection algorithm with the same tolerance ϵ, but with much less effort.

Implemented Population Monte Carlo Algorithm A main difficulty when using a PMC algorithm is the choice of the decreasing threshold values δ_t (see Section 6.2.4 "Population Monte Carlo" on page 57 for the basic algorithm). The PMC algorithm implemented in `ABCsampler` differs slightly from the one published by Beaumont et al. (2009) in order to have some sort of an adaptive scheme to set the threshold δ_{ϵ_t} to use. the main idea is to adjust the threshold according to the current distribution of distances. Similar to the calibration step proposed by Wegmann and Excoffier (2008) for an MCMC without likelihood chain, Wegmann et al. (2009) proposed to specify a tolerance value ϵ and to set the threshold δ_t such that a fraction ϵ of the generated samples is used to generate the next iteration. The algorithm implemented in `ABCsampler` goes as follows:

```
PMC1 Set $t = 1$.
PMC2 For $i$=1, ..., $N_{calibration}$, repeat:
        PMC2a Draw a parameters vector $\boldsymbol{\theta}_{i,t}$ randomly from the prior.
        PMC2b Simulate $\mathcal{D}$ using $\boldsymbol{\theta}_{i,t}$.
        PMC2c Calculate summary statistics $\mathbf{s}$ on $\mathcal{D}$.
PMC3 Increment $t$.
PMC4 Calculate $\text{dist}(\mathbf{s}, \mathbf{s}_{obs})$ for all parameter vectors $\boldsymbol{\theta}_{i,t-1}$.
PMC5 Retain the $n\epsilon$ parameter vectors $\boldsymbol{\theta}_{i,t-1}$ with the smallest associated distances.
PMC6 If $t = 2$ Set all weights of the retained
```

parameter vectors $\omega_{i,t-1} = 1/N$, else set all weights

$$\omega_{i,t} \propto \frac{\pi(\boldsymbol{\theta}_{i,t})}{\sum_{j=1}^{N} \omega_{j,t-1}\varphi(\frac{1}{\tau_{t-1}^2}(\boldsymbol{\theta}_{i,t} - \boldsymbol{\theta}_{j,t-1}))}.$$

```
PMC7  Set $\tau_t^2$ as twice the weighted empirical variance of the accepted $\boldsymbol{\theta}_{i,t-1}$.
PMC8  For $i$=1, ..., $N$, repeat:
      PMC8a  Pick a parameters vector $\boldsymbol{\theta}_i^*$ from the accepted $\boldsymbol{\theta}_{i,t-1}$ with probabilities $\omega_{i,t-1}$.
      PMC8b  Generate a parameter vector $\boldsymbol{\theta}_{i,t}$ using a normal transition kernel $\boldsymbol{\theta}_{i,t}|\boldsymbol{\theta}_i^* \sim N(\boldsymbol{\theta}_i^*, \tau_{t-1}^2)$.
      PMC8c  Simulate $\mathcal{D}$ using $\boldsymbol{\theta}_{i,t}$.
      PMC8d  Calculate summary statistics $\mathbf{s}$ on $\mathcal{D}$.
PMC9  Go back to PMC3
```

Again, $\varphi()$ denotes the multivariate normal density function. Note also that the first iteration may have a different sample size from the following iterations ($N_{calibration} \neq N$). The number of retained simulations, however, is kept constant.

6.3 Post–sampling Adjustment

If we suppose that the likelihood $f_{\mathcal{M}}(\mathbf{s}|\boldsymbol{\theta})$ of the full model is continuous and non-zero around $\mathbf{s}_{obs}$ then the likelihood of a truncated model $\mathcal{M}_\epsilon(\mathbf{s}_{obs})$ obtained by any of the rejection sampling methods mentioned above is given by

$$f_\epsilon(\mathbf{s}|\boldsymbol{\theta}) = \frac{\text{Ind}(\mathbf{s} \in \mathcal{B}_\epsilon(\mathbf{s}_{obs})) \cdot f_{\mathcal{M}}(\mathbf{s}|\boldsymbol{\theta})}{\int_{\mathcal{B}_\epsilon} f_{\mathcal{M}}(\mathbf{s}|\boldsymbol{\theta})d\mathbf{s}}, \tag{17}$$

where $\mathcal{B}_\epsilon = \mathcal{B}_\epsilon(\mathbf{s}_{obs}) = \{\mathbf{s} \in \mathbb{R}^n | \text{dist}(\mathbf{s}, \mathbf{s}_{obs}) < \epsilon\}$ is the ϵ-ball in the space of summary statistics and Ind$(\cdot)$ is the indicator function. Observe that $f_\epsilon(\mathbf{s}|\boldsymbol{\theta})$ degenerates to a (Dirac) point measure centered at $\mathbf{s}_{obs}$ as $\epsilon \to 0$. If the parameters are generated from a prior $\pi(\boldsymbol{\theta})$, then the distribution of the parameters retained after the rejection process outlined above is given by

$$\pi_\epsilon(\boldsymbol{\theta}) = \frac{\pi(\boldsymbol{\theta}) \int_{\mathcal{B}_\epsilon} f_{\mathcal{M}}(\mathbf{s}|\boldsymbol{\theta})d\mathbf{s}}{\int_\Pi \pi(\boldsymbol{\theta}) \int_{\mathcal{B}_\epsilon} f_{\mathcal{M}}(\mathbf{s}|\boldsymbol{\theta})d\mathbf{s}d\boldsymbol{\theta}}. \tag{18}$$

We shall call this density the *truncated prior.* .It is not hard to check that

$$\begin{aligned}\pi(\boldsymbol{\theta}|\mathbf{s}_{obs}) &= \frac{f_{\mathcal{M}}(\mathbf{s}_{obs}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\int_\Pi f_{\mathcal{M}}(\mathbf{s}_{obs}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}} \\ &= \frac{f_\epsilon(\mathbf{s}_{obs}|\boldsymbol{\theta})\pi_\epsilon(\boldsymbol{\theta})}{\int_\Pi f_\epsilon(\mathbf{s}_{obs}|\boldsymbol{\theta})\pi_\epsilon(\boldsymbol{\theta})d\boldsymbol{\theta}}.\end{aligned} \tag{19}$$

Thus the posterior distribution of the model $\mathcal{M}$ for $\mathbf{s} = \mathbf{s}_{obs}$ given the prior $\pi(\boldsymbol{\theta})$ is exactly equal to the posterior distribution of the truncated model $\mathcal{M}_\epsilon(\mathbf{s}_{obs})$ given the truncated prior $\pi_\epsilon(\boldsymbol{\theta})$.

In a rejection sampling framework, the empirical distribution of the accepted parameters $\mathcal{P} = \{\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^N\}$ yields an estimate of the truncated prior $\pi_\epsilon(\boldsymbol{\theta})$. To obtain an approximation of the posterior $\pi(\boldsymbol{\theta}|\mathbf{s}_{obs})$ it is implicitly assumed that the likelihood $f_{\mathcal{M}}(\mathbf{s}|\boldsymbol{\theta})$ is close to some (unknown) constant over the whole range of $\mathcal{B}_\epsilon(\mathbf{s}_{obs})$. Under that assumption, Equation (19) shows that $\pi(\boldsymbol{\theta}|\mathbf{s}_{obs}) \approx \pi_\epsilon(\boldsymbol{\theta})$, which is a good approximation if the tolerance ϵ is small.

However, as mentioned above, when the dimension n of summary statistics is high, the "curse of dimensionality" raises its ugly head: The tolerance must be chosen rather large or else the acceptance rate becomes prohibitively low. Since the approximation of $f_{\mathcal{M}}(\mathbf{s}|\boldsymbol{\theta})$ being constant over the whole range of $\mathcal{B}_\epsilon(\mathbf{s}_{obs})$ is unlike in this case, the precision of the approximation of the posterior distribution is distorted by the truncated prior (see Wegmann and Excoffier 2008 and Section 5.2 "Choosing Summary Statistics" on page 44). If we can make a more educated guess for a parametric statistical model of $M_\epsilon(\mathbf{s}_{obs})$, we arrive at more reasonable approximation of the posterior $\pi(\boldsymbol{\theta}|\mathbf{s}_{obs})$ even if the likelihood of the full model $\mathcal{M}$ is unknown. It is to be expected that due to the localization process the truncated model will exhibit a simpler structure than the full model $\mathcal{M}$ and thus be easier to estimate.

6.3.1 ABC–REG

As a first attempt to take the variation of $f_{\mathcal{M}}(\mathbf{s}|\boldsymbol{\theta})$ within the ϵ-ball into account, Beaumont et al. (2002) introduced a post–sampling regression adjustment (ABC-REG) of the sample $\mathcal{P}$ of retained parameters. Basically, they postulate a linear dependence between the parameters $\boldsymbol{\theta}$ and their associated summary statistics $\mathbf{s}$. More precisely, their method is based on the assumption that the sample of accepted values

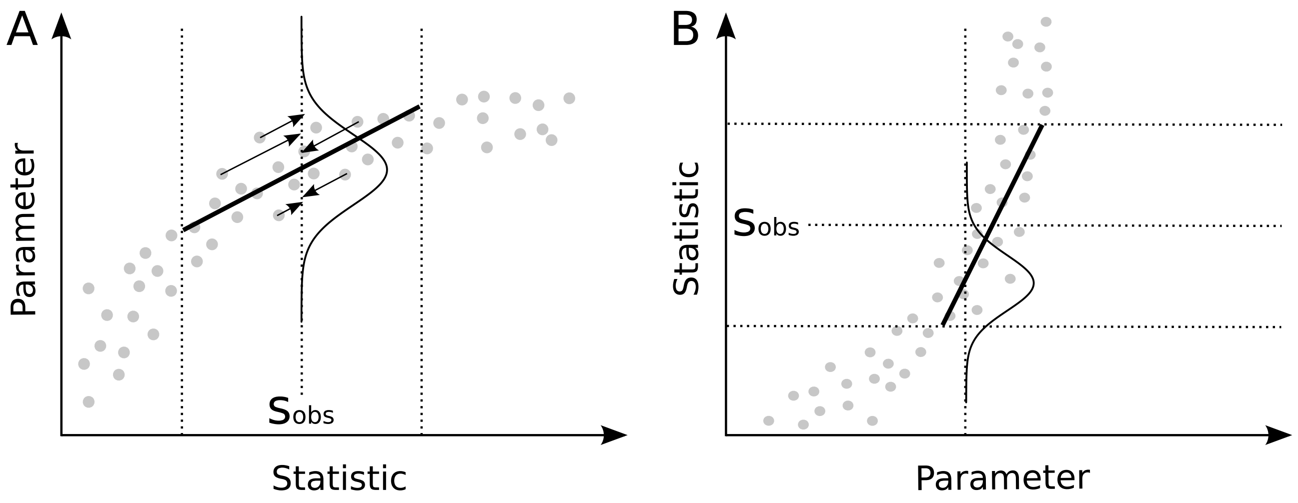


Figure 14. Post Sampling Adjustment *We discuss two different types of post sampling adjustments: (A) ABC–REG, the regression adjustment introduced by Beaumont et al. (2002) and (B), the ABC–GLM approach as described in Leuenberger and Wegmann (2009). The dotted lines around $\mathbf{s}_{obs}$ indicate the threshold used. Note that the density shown in (A) corresponds to the posterior and has an arbitrary shape, while the shown density in (B) corresponds to the likelihood at a given parameter value (dotted line) and is always normal. See text for further explanations.*

$(\boldsymbol{\theta}^j, \mathbf{s}^j)$ is approximatively generated by a linear model of the form

$$\boldsymbol{\theta} = \mathbf{M}\mathbf{s} + \mathbf{m}_0 + \boldsymbol{\epsilon}, \tag{20}$$

where $\mathbf{M}$ is a matrix of regression coefficients, $\mathbf{m}_0$ a constant vector and $\boldsymbol{\epsilon}$ a random vector of zero mean. Beaumont et al. (2002) proposed to use this linear relationship to estimate the posterior distribution at $\mathbf{s}_{obs}$ as the observed density of the expected parameters. The principle is outlined in Figure 14A and described in the following: Suppose the relationship between a parameter and a statistic of an arbitrary model follows the distribution shown in Figure 14A, where individual dots represent individual simulations. In a first step we reject all simulations, which are outside an arbitrary tolerance ϵ, as indicated by the dotted lines. While the overall relationship between the statistic and the parameter is clearly non-linear, a linear approximation may apply locally within the boundary of the tolerance. Beaumont et al. (2002) propose to use this linear relationship in the following way: firstly, the relationship is estimated via a local linear regression within the tolerance boundaries. This relationship is now used to estimate the expected parameter value leading to $\mathbf{s}_{obs}$ for each simulation. Since the estimated regression assumes homoscedacity, i.e. constant variance along the regression, the expected parameter value leading to $\mathbf{s}_{obs}$ for a given simulation can be estimated by projecting the parameter / statistic pair along the regression to $\mathbf{s}_{obs}$ (indicated with arrows in Figure 14A). The posterior distribution is then obtained by a kernel density estimation on these expected parameter values. Since the fit of the regression may be better locally around $\mathbf{s}_{obs}$ than at the tolerance boundary, Beaumont et al. (2002) further propose to weight the samples by a function of the distance of a simulation to $\mathbf{s}_{obs}$ using an Epanechnikov kernel.

Computer simulations suggest that for many population genetic models this approach (termed ABC-REG) yields posterior marginal densities that have narrower HPD (highest posterior density) regions and are more closely centered around the true parameter values than the empirical posterior densities directly produced by ABC-samplers (Beaumont et al. 2002; Wegmann and Excoffier 2008). A further advantage of these regression-based methods is their easy software implementation and their numerical stability. Note that more sophisticated regression models may be invoked. For instance, Blum and Francois (2009) proposed to use a nonlinear, heteroscedastic regression where the parameters are estimated via neuronal networks.

6.3.2 ABC-GLM

As noted recently (Leuenberger and Wegmann 2009), it would however be more natural to take the parameters $\boldsymbol{\theta}$ as exogenous and the summary statistics $\mathbf{s}$ as endogenous variables of the linear model. As a consequence, ABC-REG does not take the prior distribution into account; it can lead to grossly misspecified posteriors, in particular, it often yields posteriors that are non-zero in parameter regions, where the priors actually vanish (see Leuenberger and Wegmann 2009 for an illustration of this phenomenon).

As an alternative, a methodology (termed ABC-GLM) similar in spirit to ABC-REG but closer to the standard Bayesian setting has been proposed (Leuenberger and Wegmann 2009). In their setting, the accepted data is considered as being produced by a General Linear Model (abbreviated as GLM in the literature – not to be confused with the Generalized Linear Models which unfortunately share the same abbreviation). The model $\mathcal{M}_{GLM}$ is assumed to be normal linear, i.e. the random vectors $\mathbf{s}$ satisfy

$$\mathbf{s}|\boldsymbol{\theta} = \mathbf{C}\boldsymbol{\theta} + \mathbf{c}_0 + \boldsymbol{\epsilon}, \qquad (21)$$

where $\mathbf{C}$ is a $n \times m$-matrix of constants, $\mathbf{c}_0$ a $n \times 1$-vector and $\boldsymbol{\epsilon}$ a random vector with a multivariate normal distribution of zero mean and covariance matrix $\boldsymbol{\Sigma}_s$:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_s). \qquad (22)$$

$\mathbf{C}$, $\mathbf{c}_0$ and $\boldsymbol{\Sigma}_s$ are estimated by standard multivariate regression analysis from the sample $\mathcal{P}$, $\mathcal{S}$ created by an ABC sampler mentioned above, similar to ABC-REG. The covariance matrix encapsulates the strong correlations normally present between the components of the summary statistics, which is an advantage of GLMs.

The ABC-GLM approach is outlined in Figure 14B. Assume the same situation as for the ABC-REG approach shown in Figure 14A. The first step is, again, the rejection of simulations outside an arbitrary tolerance ϵ. Then, the parameters of the GLM are estimated on the retained samples. Note that the estimated likelihood function of the statistical model at a given parameter value $\boldsymbol{\theta}_i$ necessarily follows the normal distribution $f(s|\boldsymbol{\theta}_i \sim \mathcal{N}(E(s|\boldsymbol{\theta}_i), \boldsymbol{\Sigma}_s)$, where $E(s|\boldsymbol{\theta}_i)$ is the expected value of s at $\boldsymbol{\theta}_i$ and $\boldsymbol{\Sigma}_s$ the covariance matrix of the error term $\boldsymbol{\epsilon}$, both estimated by the standard multivariate regression (see above). Under the assumption that the statistical model within the $\boldsymbol{\epsilon}$–ball is true, the true posterior distribution is achieved when this likelihood function of the statistical model is multiplied with the truncated prior distribution, as was shown by Leuenberger and Wegmann (2009).

Of course, the use of a post–sampling adjustment is not limited to a particular ABC sampler but can be applied to all sorts of samplers, including MCMC without likelihoods (Marjoram et al. 2003) and Population Monte Carlo (Beaumont et al. 2009). The only requirement is that an ABC sampler produces samples from an empirical posterior distribution of the form $\pi(\boldsymbol{\theta}|\text{dist}(\mathbf{s}, \mathbf{s}_{obs}) < \epsilon)$. Also, more sophisticated regression schemes may be implemented within the ABC–GLM as well as within the ABC-REG framework (e.g Blum and Francois 2009).

Details of the Posterior Estimation As mentioned above, the true posterior of the initial model $\mathcal{M}$ is exactly equal to the posterior distribution of the truncated model $c_\epsilon(\mathbf{s}_{obs})$, given the truncated prior $\pi_\epsilon(\boldsymbol{\theta})$ (see Section 6.3 "Post–sampling Adjustment" on page 58). If we can estimate the truncated prior and make an educated guess for a parametric statistical model of $\mathcal{M}_\epsilon(\mathbf{s}_{obs})$, we arrive at a reasonable approximation of the posterior $\pi(\boldsymbol{\theta}|\mathbf{s}_{obs})$, even if the likelihood of the full model $\mathcal{M}$ is unknown. It is to be expected that due to the localization process the truncated model will exhibit a simpler structure than the full model $\mathcal{M}$ and thus be easier to estimate.

To fix the notation, let $\mathcal{P} = \{\boldsymbol{\theta}^1, \dots, \boldsymbol{\theta}^N\}$ be a sample of vector-valued parameters created by some ABC-algorithm simulating from some prior $\pi(\boldsymbol{\theta})$ and $\mathcal{S} = \{\mathbf{s}^1, \dots, \mathbf{s}^N\}$ the sample of associated summary statistics produced by the model $\mathcal{M}$. Each parameter $\boldsymbol{\theta}^j$ is an m-dimensional column vector $\boldsymbol{\theta}^j = (\theta^j, \dots, \theta^j_m)^T$ and each summary statistics an n-dimensional column vector $\mathbf{s}^j = (s^j_1, \dots, s^j_n)^T \in \mathcal{B}_\epsilon(\mathbf{s}_{obs})$. The samples $\mathcal{P}$ and $\mathcal{S}$ can thus be viewed as $m \times N$- and $n \times N$-matrices $\mathbf{P}$ and $\mathbf{S}$, respectively.

The empirical estimate of the truncated prior $\pi_\epsilon(\boldsymbol{\theta})$ is given by the discrete distribution that puts a point mass of $1/N$ on each value $\boldsymbol{\theta}^j \in \mathcal{P}$. We smoothen out this empirical distribution by placing a sharp Gaussian peak over each parameter value $\boldsymbol{\theta}^j$. More precisely, we set

$$\pi_\epsilon(\boldsymbol{\theta}) = \frac{1}{N}\sum_{j=1}^{N}\phi(\boldsymbol{\theta}-\boldsymbol{\theta}^j, \boldsymbol{\Sigma}_\theta), \tag{23}$$

where

$$\phi(\varrho, \boldsymbol{\Sigma}_\theta) = \frac{1}{|2\pi\boldsymbol{\Sigma}_\theta|^{\frac{1}{2}}}\exp\left(-\frac{\varrho^T\boldsymbol{\Sigma}_\theta^{-1}\varrho}{2}\right) \tag{24}$$

with

$$\varrho = (\boldsymbol{\theta}-\boldsymbol{\theta}^j) \tag{25}$$

and

$$\boldsymbol{\Sigma}_\theta = \mathrm{diag}(\sigma_1, \ldots, \sigma_m) \tag{26}$$

is the covariance matrix of ϕ, which determines the width of the Gaussian peaks. The larger the number N of sampled parameter values, the sharper the peaks can be chosen in order to still get a rather smooth π_ϵ. Too small values of σ_k will result in wiggly posterior curves, too large values might unduly smear out the curves. The best advice is to run the calculations with several choices for $\boldsymbol{\Sigma}_\theta$.

Leuenberger and Wegmann (2009) assumed the truncated model $\mathcal{M}_\epsilon(\mathbf{s}_{obs})$ to be normal linear (i.e. to be a General Linear Model GLM). To be precise, we assume the summary statistics $\mathbf{s}$, created by the truncated model's likelihood $f_\epsilon(\mathbf{s}|\boldsymbol{\theta})$, to satisfy

$$\mathbf{s}|\boldsymbol{\theta} = \mathbf{C}\boldsymbol{\theta} + \mathbf{c}_0 + \boldsymbol{\epsilon}, \tag{27}$$

where $\mathbf{C}$ is a $n \times m$-matrix of constants, $\mathbf{c}_0$ a $n \times 1$-vector and $\boldsymbol{\epsilon}$ a random vector with a multivariate normal distribution of zero mean and covariance matrix $\boldsymbol{\Sigma}_s$:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_s). \tag{28}$$

The covariance matrix $\boldsymbol{\Sigma}_s$ encapsulates the strong correlations normally present between the components of the summary statistics. `ABCestimator` estimates $\mathbf{C}$, $\mathbf{c}_0$ and $\boldsymbol{\Sigma}_s$ by standard multivariate regression analysis from the sample $\mathcal{P}$, $\mathcal{S}$[4][5].

The likelihood for this model – dropping the hats on the matrices to simplify the notation – is given by

$$f_\epsilon(\mathbf{s}|\boldsymbol{\theta}) = |2\pi\boldsymbol{\Sigma}_s|^{-\frac{1}{2}} \cdot \exp\left(-\frac{\gamma^T\ \boldsymbol{\Sigma}_s^{-1}\ \gamma}{2}\right) \tag{29}$$

with

$$\gamma = (\mathbf{s} - \mathbf{C}\boldsymbol{\theta} - \mathbf{c}_0). \tag{30}$$

Recall from (19) that for a prior $\pi(\boldsymbol{\theta})$ and an observed summary statistics $\mathbf{s}_{obs}$, the parameter's posterior distribution for our full model $\mathcal{M}$ is given by

$$\pi(\boldsymbol{\theta}|\mathbf{s}_{obs}) = c \cdot f_\epsilon(\mathbf{s}_{obs}|\boldsymbol{\theta})\pi_\epsilon(\boldsymbol{\theta}), \tag{31}$$

where $f_\epsilon(\mathbf{s}_{obs}|\boldsymbol{\theta})$ is the likelihood of the truncated model $\mathcal{M}_\epsilon(\mathbf{s}_{obs})$ given by (29) and $\pi_\epsilon(\boldsymbol{\theta})$ is the estimated (and smoothed) truncated prior given by (23).

Leuenberger and Wegmann (2009) showed that the posterior is – up to a multiplicative constant – of the form $\sum_{i=j}^{N}\exp(-\frac{1}{2}Q_j)$, where

$$\begin{aligned} Q_j \quad = \quad & (\boldsymbol{\theta}-\mathbf{t}^j)^T\mathbf{T}^{-1}(\boldsymbol{\theta}-\mathbf{t}^j) + \ldots \\ & \ldots + (\mathbf{s}_{obs}-\mathbf{c}_0)^T\boldsymbol{\Sigma}_s^{-1}(\mathbf{s}_{obs}-\mathbf{c}_0) + \ldots \\ & \ldots + (\boldsymbol{\theta}^j)^T\boldsymbol{\Sigma}_\theta^{-1}\boldsymbol{\theta}^j - (\mathbf{v}^j)^T\mathbf{T}\mathbf{v}^j. \end{aligned} \tag{32}$$

Here $\mathbf{T}, \mathbf{t}^j$ and $\mathbf{v}^j$ are given by

$$\mathbf{T} = \left(\mathbf{C}^t\boldsymbol{\Sigma}_s^{-1}\mathbf{C} + \boldsymbol{\Sigma}_\theta^{-1}\right)^{-1} \tag{33}$$

and $\mathbf{t}^j = \mathbf{T}\mathbf{v}^j$, where

$$\mathbf{v}^j = \mathbf{C}^t\boldsymbol{\Sigma}_s^{-1}(\mathbf{s}_{obs}-\mathbf{c}_0) + \boldsymbol{\Sigma}_\theta^{-1}\boldsymbol{\theta}^j. \tag{34}$$

From this we get

$$\pi(\theta|\mathbf{s}_{obs}) \propto \sum_{j=1}^{N} c(\boldsymbol{\theta}^j)\exp\left(-\frac{(\boldsymbol{\theta}-\mathbf{t}^j)^T\mathbf{T}^{-1}(\boldsymbol{\theta}-\mathbf{t}^j)}{2}\right), \tag{35}$$

where

$$c(\boldsymbol{\theta}^i) = \exp\left[-\frac{(\boldsymbol{\theta}^j)^t\boldsymbol{\Sigma}_\theta^{-1}\boldsymbol{\theta}^j - (\mathbf{v}^j)^T\mathbf{T}\mathbf{v}^j}{2}\right]. \tag{36}$$

Marginal posterior density of the parameter θ_k

[4] We refer the reader to the original paper by Leuenberger and Wegmann (2009).

[5] Strictly speaking, one must redo an ABC-sample from uniform priors over Π in order to get an unbiased estimate of the GLM if the prior $\pi(\boldsymbol{\theta})$ is not uniform already. On the other hand, ordinary least squares estimators are quite insensitive to the prior's influence. In practice one can as well use the sample $\mathcal{P}$ to the estimate.

defined by

$$\pi(\theta_k|\mathbf{s}) = \int_{\mathbb{R}^{m-1}} \pi(\boldsymbol{\theta}|\mathbf{s}) d\boldsymbol{\theta}_{-k}, \tag{37}$$

where integration is performed along all parameters except θ_k.

Recall that the marginal distribution of a multivariate normal $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to the k-th component is the univariate normal density $\mathcal{N}(\mu_k, \sigma_{k,k})$. Using this fact, Leuenberger and Wegmann (2009) showed that the marginal posterior of parameter $\boldsymbol{\theta}_k$ is given by

$$\pi(\theta_k|\mathbf{s}_{obs}) \propto \sum_{j=1}^{N} c(\boldsymbol{\theta}^j) \exp\left(-\frac{(\theta_k - t_k^j)^2}{2\tau_{k,k}}\right), \tag{38}$$

where $\tau_{k,k}$ is the k-th diagonal element of the matrix $\mathbf{T}$, t_k^j is the k-th component of the vector $\mathbf{t}^j$, and $c(\boldsymbol{\theta}^j)$ is still determined according to (Equation 36). The normalizing constant is recovered through a numerical integration.

6.4 Model Choice

For two models $\mathcal{M}_A$ and $\mathcal{M}_B$ with prior probabilities π_A and $\pi_B = 1 - \pi_A$, the Bayes factor B_{AB} in favor of model $\mathcal{M}_A$ over model $\mathcal{M}_B$ is

$$B_{AB} = \frac{f_{\mathcal{M}_A}(\mathbf{s}_{obs})}{f_{\mathcal{M}_B}(\mathbf{s}_{obs})}, \tag{39}$$

where the marginal densities $f_{\mathcal{M}_A}$ and $f_{\mathcal{M}_B}$ are given by

$$f_{\mathcal{M}}(\mathbf{s}_{obs}) = \int_{\Pi} f(\mathbf{s}_{obs}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{40}$$

According to Leuenberger and Wegmann (2009) the marginal density is also given by

$$f_{\mathcal{M}}(\mathbf{s}_{obs}) = A_\epsilon(\mathbf{s}_{obs}, \pi) \int_{\Pi} f_\epsilon(\mathbf{s}_{obs}|\boldsymbol{\theta})\pi_\epsilon(\boldsymbol{\theta}) d\boldsymbol{\theta}, \tag{41}$$

where $A_\epsilon(\mathbf{s}_{obs}, \pi)$ is the acceptance rate of the rejection step. Using the parameters of the GLM estimated in the posterior estimation step (see above), the marginal density is given by:

$$f_{\mathcal{M}}(\mathbf{s}_{obs}) = \frac{A_\epsilon(\mathbf{s}_{obs}, \pi)}{N|2\pi\mathbf{D}|^{1/2}} \cdot \ldots$$
$$\ldots \sum_{j=1}^{N} \exp\left(-\frac{\zeta^T \mathbf{D}^{-1} \zeta}{2}\right), \tag{42}$$

where

$$\zeta = (\mathbf{s}_{obs} - \mathbf{m}^j) \tag{43}$$

and the sum runs over the parameter sample $\mathcal{P} = \{\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^N\}$,

$$\mathbf{D} = \boldsymbol{\Sigma}_s + \mathbf{C}\boldsymbol{\Sigma}_\theta \mathbf{C}^T \tag{44}$$

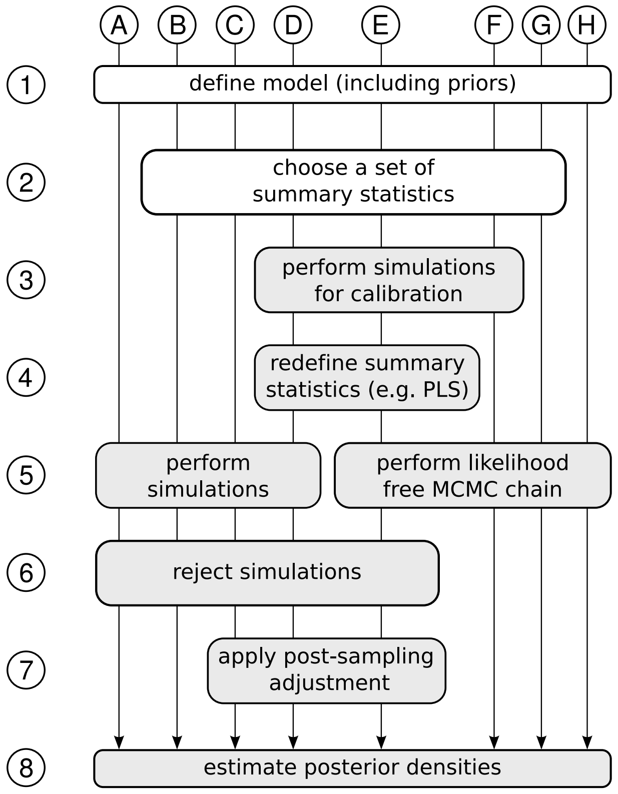
and

$$\mathbf{m}^j = \mathbf{c}_0 + \mathbf{C}\boldsymbol{\theta}^j. \tag{45}$$

6.5 Likelihood free without approximations

Likelihood free samplers were introduced to allow Bayesian inference in situations where the likelihood $f_{\mathcal{M}}(\mathcal{D}|\boldsymbol{\theta})$ is analytically intracable (Beaumont et al. 2002; Marjoram et al. 2003; Sisson et al. 2007; Tavare et al. 1997). However, none of these samplers requires any approximation explicitly. All samplers have been proven to converge to the desired posterior distribution $\pi(\boldsymbol{\theta}|\mathcal{D})$, if run for long enough and if tolerance $\epsilon = 0$ (Marjoram et al. 2003; Sisson et al. 2007; Tavare et al. 1997). The approximations discussed above rather make the application of the different samplers more general. While it is in many cases possible to run these samplers with a tolerance $\epsilon = 0$ and with comparing $\mathcal{D}'$ with $\mathcal{D}$ directly, this is often not desirable. Firstly, as portions of the data are usually not influenced by the parameters in question (see Section 1.1.3 "Summary Statistics" on page 5), the acceptance rate is too low. This may be circumvented by using sufficient summary statistics, which will by itself not lead to an approximated posterior distribution. Secondly, the probability to observe $\mathcal{D}' = \mathcal{D}$ is very low. In order to deal with that, tolerance values $\epsilon > 0$ are usually used, leading to an obligatory approximation.

Thus, running likelihood free samplers on the full data $\mathcal{D}$ or by using sufficient summary statistics in combination with a tolerance $\epsilon = 0$ is not considered an approximate approach. Note that even if tolerance values $\epsilon > 0$ are used, but the relationship between parameters $\boldsymbol{\theta}$ and summary statistics $\mathbf{s}$ of the model match the assumptions of the post–sampling adjustment used exactly, approximation free inferences are possible. This is tough, unfortunately, a very unlikely situation.



A
B
C
D
E
F
G
H
1
define model (including priors)
2
choose a set of summary statistics
3
perform simulations for calibration
4
redefine summary statistics (e.g. PLS)
5
perform simulations
perform likelihood free MCMC chain
6
reject simulations
7
apply post-sampling adjustment
8
estimate posterior densities

Figure 15. ABC wokflow *The different ABC procedures proposed are shown as vertical arrows. Any ABC estimation involves several steps outlines as boxes here, where grey boxes show steps for which this package provides software. An ABC estimation always starts with the definition of a model and the priors of the parameters of this model. While all ABC procedures share the goal of estimating the posterior densities, the involved steps to get there may vary considerably. See text for a discussion of the different procedures.*

6.6 An ABC Flowchart

In this section we outline the workflow of different ABC procedures (see Figure 15). While the methodological details of these different algorithms are given above, we aim at clarifying the different steps involved in ABC estimations, as well as highlighting the differences of the procedures proposed. Please refer to the original publications (see below) for details on the different algorithms.

In any case, the definition of a model of interest, including all priors for the parameters of the model, marks the first step. The simplest procedures are those, where the simulated data $\mathcal{D}'$ is compared with the observed data $\mathcal{D}$ directly (Ⓐ and Ⓗ in Figure 15). These have been proposed by Tavare et al. (1997) and Marjoram et al. (2003), respectively. However, due to reasons discussed above, none of these algorithms have been used in population genetics so far. In order to run their algorithm successfully, both, Tavare et al. (1997) and Marjoram et al. (2003), approximated the full data with summary statistics, which resulted in algorithms Ⓑ and Ⓖ, respectively. Beaumont et al. (2002) proposed to deal with the often relatively large threshold values needed in the rejection step ⑥ by applying a post–sampling regression adjustment ⑦, therefore proposing algorithm Ⓒ. Later, Wegmann and Excoffier (2008) proposed to combine the advantages of likelihood free MCMC inference ⑤ with the benefits of applying post–sampling adjustments ⑥. Additionally, they proposed to perform calibration simulations ③ prior to the launch of an MCMC chain, dealing that way with difficulties of the likelihood free MCMC, which is often prone to get stuck or to sit in the tails of the distribution (Sisson et al. 2007). Additionally, these calibration simulations may be used to lower the dimensionality of the summary statistics space, as proposed by Wegmann and Excoffier (2008), who used a PLS transformation to achieve this, resulting is algorithm Ⓔ. Another way to speed up the likelihood free MCMC is to use a tempering scheme on the threshold, as proposed by Ratmann et al. (2007), who used algorithm Ⓖ. While only published very recently (Beaumont et al. 2009), the PMC algorithm can be used just as the likelihood free MCMC.

If a considered model is very fast to simulate or if the same simulations are to be used for different observed data sets, it may be easier and faster to stick to a pure rejection sampling algorithm. Of course, the reduction of the summary statistics space is also advisable in this case (see Section 5.2 "Choosing Summary Statistics" on page 44). In these cases, algorithm Ⓓ may be applied. Note that the calibration simulations may easily be used for posterior estimation in this case.

7 Acknowledgments

We thank Miriam Schädler for insightful comments on an earlier draft of this manual and Benjamin Peter for his valid feedbacks as a beta tester. This work has been supported by a grant of the Swiss National Foundation No 3100A0-112072 to L.E.

8 References

Beaumont, M. A., W. Zhang, and D. J. Balding (2002). "Approximate bayesian computation in population genetics". In: *Genetics* 162.4. 22412157 0016-6731 Journal Article. Pp. 2025–35.

Beaumont, M, J-M Cornuet, J-M Marin, and C Robert (2009). "Adaptivity for approximate Bayesian computation algorithms: a population Monte Carlo approach". In: *Biometrika*.

Blum, M. and O. Francois (2009). "Non-linear regression models for Approximate Bayesian Computation". In: *Arxiv*.

Boulesteix, A. L. and K. Strimmer (2007). "Partial least squares: a versatile tool for the analysis of high-dimensional genomic data". In: *Briefings in Bioinformatics* 8.1. Journal Article Review England. Pp. 32–44.

Chadeau-Hyam, Marc et al. (2008). "Fregene: Simulation of realistic sequence-level data in populations and ascertained samples". In: *BMC Bioinformatics* 9.1. P. 364. ISSN: 1471-2105. DOI: `10.1186/1471-2105-9-364`. URL: `http://www.biomedcentral.com/1471-2105/9/364`.

Cook, S, A Gelman, and D Rubin (2006). "Validation of Software for bayesian Models Using Posterior Quantiles". In: *Journal of Computational and Graphical Statistics* 15.3. Pp. 675–692.

Cornuet, Jean-Marie et al. (2008). "Inferring population history with DIY ABC: a user-friendly approach to approximate Bayesian computation". In: *Bioinformatics* 24.23. Pp. 2713–2719. DOI: `10.1093/bioinformatics/btn514`. URL: `http://bioinformatics.oxfordjournals.org/cgi/content/abstract/24/23/2713`.

Excoffier, L., A. Estoup, and J. M. Cornuet (2005). "Bayesian analysis of an admixture model with mutations and arbitrarily linked markers". In: *Genetics* 169.3. Pp. 1727–1738.

Excoffier, Laurent, J. Novembre, and Stefan Schneider (2000). "SIMCOAL: A general coalescent program for the simulation of molecular data in interconnected populations with arbitrary demography". In: *The Journal of Heredity* 91. Laurent. Pp. 506–510.

Hey, J. and R. Nielsen (2004). "Multilocus methods for estimating population sizes, migration rates and divergence time, with applications to the divergence of Drosophila pseudoobscura and D-persimilis". In: *Genetics* 167.2. Pp. 747–760.

— (2007). "Integration within the Felsenstein equation for improved Markov chain Monte Carlo methods in population genetics". In: *Proceedings of the National Academy of Sciences of the United States of America* 104.8. Pp. 2785–2790.

Hudson, R. R. (2002). "Generating samples under a Wright-Fisher neutral model of genetic variation". In: *Bioinformatics* 18.2. 21835528 1367-4803 Journal Article. Pp. 337–8.

Joyce, P and P Marjoram (2008). "Approximately Sufficient Statistics and Bayesian Computation". In: *Statistical Applications in Genetics and Molecular Biology* 7.1.

Kingman, J. F. C. (1982). "The coalescent". In: *Stochastic Processes and their Applications* 13. Gerald Laurent. Pp. 235–248.

Kingman, J.F.C. (1982). "On the Genealogy of Large Populations". In: *Advances in Applied Probability*. Pp. 27–43.

Laval, G. and L. Excoffier (2004). "SIMCOAL 2.0: a program to simulate genomic diversity over large recombining regions in a subdivided population with a complex history". In: *Bioinformatics* 20.15. Pp. 2485–2487.

Leuenberger, Christoph and Daniel Wegmann (2009). "Bayesian Computation and Model Selection in Population Genetics". In: *genetics*.

Marjoram, P. and S. Tavare (2006). "Modern computational approaches for analysing molecular genetic variation data". In: *Nature Reviews Genetics* 7.10. Pp. 759–770.

Marjoram, P., J. Molitor, V. Plagnol, and S. Tavare (2003). “Markov chain Monte Carlo without likelihoods”. In: *Proceedings Of The National Academy Of Sciences Of The United States Of America* 100.26. Pp. 15324–15328.

Mevik, B. H. and R. Wehrens (2007). “The pls package: Principal component and partial least squares regression in R”. In: *Journal of Statistical Software* 18.2.

Neuenschwander, Samuel, Frederic Hospital, Frederic Guillaume, and Jerome Goudet (2008). “quantiNemo: an individual-based program to simulate quantitative traits with explicit genetic architecture in a dynamic metapopulation”. In: *Bioinformatics* 24.13. Pp. 1552–1553. DOI: `10.1093/bioinformatics/btn219`. URL: `http://bioinformatics.oxfordjournals.org/cgi/content/abstract/24/13/1552`.

Ratmann, O. et al. (2007). “Using likelihood-free inference to compare evolutionary dynamics of the protein networks of H. pylori and P. falciparum”. In: *Plos Computational Biology* 3.11. ISI Document Delivery No.: 236NP Times Cited: 0 Cited Reference Count: 57. Pp. 2266–2278.

Sisson, S. A., Y. Fan, and M. M. Tanaka (2007). “Sequential Monte Carlo without likelihoods”. In: *Proceedings of the National Academy of Sciences of the United States of America* 104.6. Pp. 1760–1765.

Tavare, S., D. J. Balding, R. C. Griffiths, and P. Donnelly (1997). “Inferring coalescence times from DNA sequence data”. In: *Genetics* 145.2. 0016-6731 Journal Article. Pp. 505–18.

Tenehaus, M., J.-P. Gauchi, and C. Ménardo (1995). “Régression pls et applications”. In: *Rev. Statist. Appl.* 43.1. P. 57.

Thalmann, O. et al. (2009). “A genetic perspective on the recent past and uncertain future of the critically endangered Cross River Gorilla (Gorilla gorilla diehli)”. In: *somewhere*.

Wegmann, D. and L. Excoffier (2008). “Efficient Approximate Bayesian Computation coupled with Markov Chain Monte Carlo without likelihood”. In: *Genetics*.

Wegmann, D., C. Leuenberger, S. Neuenschwander, and L. Excoffier (2009). “ABCbox1.0: A Toolkit to perform various ABC Algorithms”. In: *Bioinformatics*.

Weiss, G. and A. von Haeseler (1998). “Inference of population history using a likelihood approach”. In: *Genetics* 149.3. Pp. 1539–1546. ISSN: 0016-6731.

Index