abctools: An R Package for Tuning Approximate Bayesian Computation Analyses

by Matthew A. Nunes and Dennis Prangle

Abstract Approximate Bayesian computation (ABC) is a popular family of algorithms which perform approximate parameter inference when numerical evaluation of the likelihood function is not possible but data can be simulated from the model. They return a sample of parameter values which produce simulations close to the observed dataset. A standard approach is to reduce the simulated and observed datasets to vectors of *summary statistics* and accept when the difference between these is below a specified threshold. ABC can also be adapted to perform model choice.

In this article, we present a new software package for R, **abctools** which provides methods for tuning ABC algorithms. This includes recent dimension reduction algorithms to tune the choice of summary statistics, and coverage methods to tune the choice of threshold. We provide several illustrations of these routines on applications taken from the ABC literature.

Introduction

Approximate Bayesian computation (ABC) refers to a family of statistical techniques for inference in cases where numerical evaluation of the likelihood is difficult or intractable, ruling out standard maximum likelihood and Bayesian techniques. It has been successfully applied in a wide range of scientific fields which encounter complex data and models, such as population genetics (Fagundes et al., 2007; Beaumont, 2010), ecology (Csilléry et al., 2010), infectious disease modelling (Luciani et al., 2009; Brooks-Pollock et al., 2014), systems biology (Ratmann et al., 2007; Toni et al., 2009) and astronomy (Cameron and Pettitt, 2012; Weyant et al., 2013).

ABC performs inference based on simulation of datasets rather than likelihood evaluation. For this reason it is known as a *likelihood-free* method. The simplest ABC algorithm is rejection-ABC. This simulates parameter values from the prior and corresponding datasets from the model of interest. Parameters are accepted if the distance between *summary statistics* of the simulated and the observed data is below a *threshold*, ε. A similar approach can be used to choose between several models with intractable likelihoods. In all cases two key tuning choices for ABC are ε and which summary statistics are used. **abctools** provides various tools to assist these choices. It has been designed to complement existing software for performing ABC algorithms, especially the **abc** package (Csilléry et al., 2012). The examples in this paper use version 1.0.3 of **abctools**. Note that all the methods provided require access to at least some of the datasets simulated by ABC. In this sense they are post-processing tools.

The remainder of the article is organised as follows. First a review of relevant ABC algorithms, theory and software is given. Then two data examples are introduced which will be used for illustration throughout the paper. The following section describes the summary statistic selection methods provided by **abctools**. The final section considers choice of ε using the coverage property (Prangle et al., 2014).

Review of ABC

The following algorithms perform ABC for parameter inference or model choice. This is done in a Bayesian framework. Observed data is represented by x_{obs}. One or several probability densities $p(x|\theta, m)$ are available as models for the data. Here θ is a vector of parameters and m is a model indicator. Prior model weights $p(m)$ and parameter densities for each model $p(\theta|m)$ must also be specified. (Note that there is no requirement for the length of θ to be the same in all models.) If there is only one model of interest (the parameter inference case) the model can be written as $p(x|\theta)$ and then only a single parameter prior $p(\theta)$ is needed.

The ABC algorithms require that it is possible to sample from the priors and models. They also require various tuning choices: a distance function $d(\cdot, \cdot)$ (Euclidean distance is a common choice), a threshold $\varepsilon \geq 0$ and a mapping $s(\cdot)$ from data to a vector of summary statistics.

Rejection-ABC for parameter inference

Initialisation: For the observed dataset x_{obs}, compute a vector of summary statistics $s_{\text{obs}} = s(x_{\text{obs}})$.

Main loop:

1. Draw a parameter vector $\theta' \sim p(\theta)$ from the prior distribution;

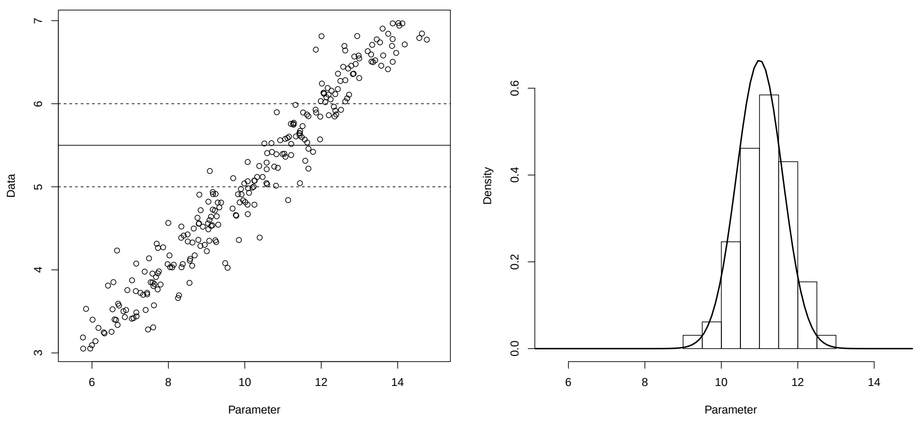

Figure 1: A pictorial illustration of the rejection-ABC algorithm for inference in a single parameter model (details omitted). The left panel shows simulated parameters and datasets. The solid horizontal line shows the observed data, and the dashed lines show acceptance boundaries. The right panel is a histogram and kernel density estimate of the accepted parameter values. These approximate the posterior distribution.

2. Generate data from the model $x' \sim p(x|\theta')$, and compute summaries $s' = s(x')$;
3. If $d(s', s_{\text{obs}}) \leq \varepsilon$, accept θ'.

Rejection-ABC for model choice

Initialisation: For the observed dataset x_{obs}, compute a vector of summary statistics $s_{\text{obs}} = s(x_{\text{obs}})$.

Main loop:

1. Draw a model $m' \sim p(m)$ from the prior distribution on models;
2. Draw a parameter vector $\theta' \sim p(\theta|m')$ from the prior distribution on parameters for model m';
3. Generate data from the model $x' \sim p(x|\theta', m')$, and compute summaries $s' = s(x')$;
4. If $d(s', s_{\text{obs}}) \leq \varepsilon$, accept (m', θ').

Both algorithms output a sample from an approximation to the posterior distribution. That is, for parameter inference the output is $\theta_1, \theta_2, \ldots$ from an approximation to $p(\theta|x_{\text{obs}})$, and for model choice the output is $(m_1, \theta_1), (m_2, \theta_2), \ldots$ from an approximation to $p(\theta, m|x_{\text{obs}})$.

If $\varepsilon = 0$ then only exact matches $x' = x_{\text{obs}}$ are accepted. It can easily be shown that in this case the output sample follows the exact posterior distribution of interest. However this is rarely practical as the probability of an exact match is typically very low for discrete data or zero for continuous data. Hence a tolerance $\varepsilon > 0$ is used, producing a sample from an approximation to the posterior (see Figure 1). An obvious acceptance criterion is $d(x', x_{\text{obs}}) \leq \varepsilon$, but this has been found to produce a poor approximation unless the data is low dimensional. Intuitively this is because close matches to the data become increasingly unlikely as the number of numerical components which must be matched increases. This *curse of dimensionality* problem motivates the use of low-dimensional summary statistics, which have greatly improved results in practice. See Beaumont (2010); Csilléry et al. (2010); Marin et al. (2012) for a more detailed discussion of this issue, and general background on ABC.

Two crucial tuning choices in rejection-ABC are the tolerance ε and the summary statistics $s(\cdot)$. Several approaches have been proposed in the literature to address these choices. **abctools** implements a range of such summary statistic selection methods and a method for choosing ε based on Prangle et al. (2014). Note that there are many other methods: for example see Blum et al. (2013) for a recent review of ABC summary statistic selection methods and Csilléry et al. (2012) for choice of ε by cross-validation.

There are several ABC algorithms which are more efficient than rejection-ABC. These concentrate on simulating from models and parameter values close to previously successful values. These include Markov chain Monte Carlo (Marjoram et al., 2003; Sisson and Fan, 2011) and sequential Monte Carlo (SMC) techniques (Sisson et al., 2007; Toni et al., 2009; Beaumont et al., 2009; Del Moral et al., 2012). A complementary approach is to post-process ABC output to reduce the approximation in using $\varepsilon > 0$ by adjusting accepted parameter values via regression onto the observed summary statistics (Beaumont et al., 2002; Blum and François, 2010). In both cases tuning ε and $s(\cdot)$ remains of crucial importance. All of the **abctools** methods can be used with post-processing. Also, all of the summary statistic selection methods can be adapted for use with other ABC algorithms and the details of this are discussed below. However the approach to tuning ε is applicable to rejection-ABC only. The reason is that ABC must be repeated under many different observations, and this is

Name	References	Stand-alone	Platform	Models
abc	Csilléry et al. (2012)	No (R package)	All	General
ABCreg	Thornton (2009)	Yes	Linux, OS X	General
easyABC	Jabot et al. (2013)	No (R package)	All	General
ABCtoolbox	Wegmann et al. (2010)	Yes	Linux, Windows	Genetics
Bayes-SSC	Anderson et al. (2005)	Yes	All	Genetics
DIY-ABC	Cornuet et al. (2008, 2010, 2014)	Yes	All	Genetics
msBayes	Hickerson et al. (2007)	Yes	Linux, OS X	Genetics
MTML-msBayes	Huang et al. (2011)	Yes	Linux, OS X	Genetics
onesamp	Tallmon et al. (2008)	Yes (web interface)	All	Genetics
PopABC	Lopes et al. (2009)	Yes	All	Genetics
REJECTOR	Jobin and Mountain (2008)	Yes	All	Genetics
EP-ABC	Barthelmé and Chopin (2014)	No (MATLAB toolbox)	All	State space models (and related)
ABC-SDE	Picchini (2013)	No (MATLAB toolbox)	All	Stochastic differential equations
ABC-SysBio	Liepe et al. (2010)	Yes (Python scripts)	All	Systems biology

Table 1: Software for ABC. "All" regarding platform refers to Linux, OS X (Mac) and Windows.

only computationally feasible under rejection-ABC as the same simulations can be reused each time. Some alternative methods have been proposed for the choice of ε in ABC-SMC algorithms, see for example Drovandi and Pettitt (2011); Del Moral et al. (2012); Lenormand et al. (2013).

Existing software

This section details existing software available for ABC, then outlines how **abctools** provides previously unavailable methodology and how it can be used alongside other software. Existing software is detailed in Table 1.

The software varies widely in which ABC algorithms are implemented. Of the two R packages, **abc** implements ABC-rejection with many methods of regression post-processing, while **easyABC** implements a wider suite of ABC algorithms but not post-processing. For full details of the other software see the references in Table 1.

Some of the available software packages provide methods for selecting summary statistics. A projection method based on partial least squares (Wegmann et al., 2009) is available in *ABCtoolbox*, and one for model choice based on linear discriminant analysis (Estoup et al., 2012) in *DIY-ABC*. Another category of methods is *regularisation techniques*, for example via ridge regression (Blum and François, 2010; Blum et al., 2013). Ridge regression regularisation is implemented in the R package **abc**; see Csilléry et al. (2012) for more details. The **abc** package also provide a method to choose ε by cross-validation.

The **abctools** package has been designed to complement the existing software provision of ABC algorithms by focusing on tools for tuning them. It implements many previously unavailable methods from the literature and makes them easily available to the research community. The software has been structured to work easily in conjunction with the **abc** package, but the package also has the flexibility to be used with other ABC software. This is discussed below (under "Using other ABC algorithms with **abctools**"), along with details of how the package framework can be used to implement further emerging methodology for summary statistic selection and construction.

Data examples

Summaries of genetic variation

The first dataset represents data generated from a commonly used model in population genetics. Specifically, the **abctools** package contains the two datasets `coal` and `coalobs`. The dataset `coal` is a matrix of dimension `100000 x 9`, representing parameters and summaries generated from an infinite-sites coalescent model for genetic variation (see Nordborg 2007 for more details). In particular, the parameters of interest are the scaled mutation rate, $\tilde{\theta}$, and the scaled recombination rate, ρ; columns 3–9 are data summaries, namely the number of segregating sites (C_1); the pairwise mean number of nucleotidic differences (C_3); the mean R^2 across pairs separated by $<$ 10% of the simulated genomic regions (C_4); the number of distinct haplotypes (C_5); the frequency of the most common haplotype (C_6) and the number of singleton haplotypes (C_7). The summary C_2 (column 4) is a spurious statistic, namely a standard uniform random deviate.

The data `coalobs` is a matrix of dimension `100 x 9`, representing similar instances of summary statistics from the model and associated parameters; these can be treated as observed data. Similar data were analysed in simulations in Joyce and Marjoram (2008) and Nunes and Balding (2010). The datasets can be loaded with `data(coal)` and `data(coalobs)` respectively.

A bigger dataset with 10^6 rows of similar summaries can be loaded using the code:

```
> mycon <- url("http://www.maths.lancs.ac.uk/~nunes/ABC/coaloracle.rda")
> load(mycon)
> close(mycon)
```

g-and-k distribution

The g-and-k distribution, used in various applications such as finance and environmental modelling, is a family of distributions which is specified by its quantile distribution, but does not have a closed form expression for its density (Rayner and MacGillivray, 2002). Data can easily be simulated by the inversion method. The dataset included in the **abctools** package is a matrix of dimension `100000 x 11` consisting of `n = 100000` simulations of 4 parameters (A, B, g and k), together with 7 summary statistics representing the octiles of 1000 independent draws given the corresponding parameters. Such quantiles have been used for inference in an ABC context by Drovandi and Pettitt (2011) and Fearnhead and Prangle (2012), amongst others.

The dataset can be loaded using the code:

```
> mycon <- url("http://www.maths.lancs.ac.uk/~nunes/ABC/gkdata.rda")
> load(mycon)
> close(mycon)
```

The code used to generate these simulations is available at `http://www.maths.lancs.ac.uk/~nunes/ABC/gksim.R`.

Summary statistics selection

Identifying an informative and low-dimensional set of summaries to represent high dimensional data for use in ABC methods is of high importance for meaningful posterior inference; a number of methods to achieve this have been proposed in the statistical literature. We assume there is a prespecified set of *input statistics* of the data $z(x) = \{z_1, \ldots, z_k\}$. This may be the raw data, or some transformations believed to be informative. Techniques for choosing ABC summary statistics fall into several categories, including: methods that select a *best subset* of z (Joyce and Marjoram, 2008; Nunes and Balding, 2010) and secondly, *projection techniques* that project z onto a lower dimensional space (Wegmann et al., 2009; Blum and François, 2010; Fearnhead and Prangle, 2012). A review of methods for choosing summary statistics, including those mentioned above, can be found in Blum et al. (2013). This study found that when k was relatively small, best subset methods were generally preferable, and otherwise projection techniques performed better.

In what follows we describe the implementations of a number of methods for choosing summary statistics in the **abctools** package, namely the *approximate sufficiency* algorithm of Joyce and Marjoram (2008); the entropy criterion and two-stage methods of Nunes and Balding (2010), and the *semi-automatic ABC* projection technique of Fearnhead and Prangle (2012). For summary statistics selection the user must simulate parameters and data and supply these to the package. The resulting summary statistics can then be passed to another package to perform ABC. This form of operation makes **abctools** particularly suited to rejection-ABC. Note however, that many of the main routines in this section have similar arguments, indicative of the flexible and modular nature of the package. Indeed, the final part of this section discusses the `selectsumm` wrapper function which can be used to implement any of the methods, as well as using **abctools** with other user-defined ABC routines.

Best subset methods

As outlined above, the principle of summary subset selection methods is to select a subset of informative statistics $s_A \subseteq z$ for use in ABC inference, such as the rejection-ABC algorithm described above. In this section we outline the implementations of some of these "best subset" algorithms for summary selection.

Subset selection via approximate sufficiency. Joyce and Marjoram (2008) introduced a method of summary selection based on a measure of approximate sufficiency. The idea of the sufficiency criterion is that, if a (sub)set of summaries is sufficient for θ, then adding an extra statistic won't affect the posterior distribution for θ. Motivated by this observation, the algorithm of Joyce and Marjoram (2008) sequentially tests the potential inclusion of individual statistics into the set s_A, accepting them if the change in the corresponding posterior density approximation exceeds a threshold. The change in the posterior is deemed sufficient if

$$\left| \frac{p_{ABC}\left(\theta|z_1, \ldots, z_{k-1}, z_k\right)}{p_{ABC}\left(\theta|z_1, \ldots, z_{k-1}\right)} - 1 \right| > T(\theta), \tag{1}$$

where p_{ABC} denotes a histogram estimator approximation of the posterior density. See Section 5 of Joyce and Marjoram (2008) for details of how the threshold $T(\theta)$ is defined. Note that due to the form of the criterion (1), the test is at present only suitable for single parameter inference.

The hypothesis test is performed by the **abctools** function `AS.test`. The function has inputs `x1` and `x2`, representing approximate posterior samples for the density *without* or *including* the statistic being tested, respectively. The test returns a Boolean variable (`TRUE` or `FALSE`) indicating whether the second posterior sample (as represented by `x2`) is sufficiently different from the first posterior sample `x1`.

As an example of this, running the code

```
> unif.sample <- runif(10000); norm.sample <- rnorm(10000)
> AS.test(x1 = unif.sample, x2 = norm.sample)
[1] TRUE
```

results in a statement that the two posterior samples `x1` and `x2` are judged to be statistically different.

To decide on the final set of summaries, the test is performed as a sequential search, testing candidate statistics from z in turn. The final subset s_A is dependent on the order in which statistics from z are tested for inclusion; in practice, this order is random. The sequential testing procedure is implemented in the **abctools** function `AS.select`. The main arguments of the function are:

`obs`: Input statistics corresponding to observed data, $z(x_{\text{obs}})$. This is a matrix of dimension `ndatasets x k`.

`param`: Simulated parameters (drawn from a prior) which were used to generate simulated data under the model; a matrix of dimension `nsims x p`.

`sumstats`: Input statistics $z(x)$ generated using the model with the parameters `param`; a matrix of dimension `nsims x k`.

After performing the summary search procedure, the `AS.select` function returns the final subset of statistics s_A in the `best` component of the output. if the optional `trace` argument is set to `TRUE` (the default), the function will print messages to inform the user about the summary statistics search.

An example of using the `AS.select` function using the coalescent data described above is shown below.

```
> data(coal); data(coalobs)
> param <- coal[, 2]
> simstats <- coal[, 3:9]
> obsstats <- matrix(coalobs[1, 3:9], nrow = 1)
> set.seed(1)
> AStchoice <- AS.select(obsstats, param, simstats)
Sumstat order for testing is: 2 3 6 4 1 7 5
Current subset is: empty Test adding: 2
Empty subset not allowed - add

Current subset is: 2  Test adding: 3
No significant change to ABC posterior - don't add

Current subset is: 2  Test adding: 6
No significant change to ABC posterior - don't add

Current subset is: 2  Test adding: 4
No significant change to ABC posterior - don't add

Current subset is: 2  Test adding: 1
No significant change to ABC posterior - don't add

Current subset is: 2  Test adding: 7
Significant change to ABC posterior - add

Consider removing previous summaries
Current subset is: 2 7  Test removing: 2
No significant change to ABC posterior - remove

Current subset is: 7  Test adding: 5
No significant change to ABC posterior - don't add

Selected summaries: 7
```

```
> ASchoice$best
[1] 7
```

The result of the sequential search is that out of the summary subsets tested, the single summary subset $\{C_7\}$ is judged to be the most informative.

Subset selection via minimising an information criterion. Another sequential search algorithm for summary statistics in the **abctools** package is the flexible *minimum criterion* function `mincrit`. Essentially, this function cycles through each subset of summaries in turn, and computes a specified criterion on the ABC posterior sample produced with that particular set of summaries. The best subset s_A is judged to be that which minimises the criterion over all possible subsets of statistics. The search proposed in Nunes and Balding (2010) suggests minimising the κ-nearest neighbour entropy, E of the posterior sample

$$\hat{E} = \log\left[\frac{\pi^{p/2}}{\Gamma(p/2+1)}\right] - \psi(\kappa) + \log n + \frac{p}{n}\sum_{i=1}^{n}\log R_{i,\kappa}, \tag{2}$$

where p is the dimension of the parameter vector θ, $\psi(\cdot)$ denotes the digamma function, and where $R_{i,\kappa}$ denotes the Euclidean distance from θ^i to its κ-th closest neighbour in the posterior sample (Singh et al., 2003). Nunes and Balding (2010) follow Singh et al. (2003) in using $\kappa = 4$ for reasons of numerical stability. Blum et al. (2013) extend this entropy expression for weighted posterior samples. This entropy calculation in (2) is computed in **abctools** using the `nn.ent` function. For example, for the 4th nearest neighbour entropy calculation for a posterior sample `psample`, one would use the command

```
> nn.ent(psample, k = 4)
```

The `mincrit` function has many of the same arguments as the `AS.select` function above, including `obs`, `param` and `sumstats`, see the `mincrit` function documentation in the package **abctools** for a full list. Other function arguments include `crit`, which specifies the criterion to minimise. The default for this is `nn.ent`. The heuristic for this criterion as suggested by Nunes and Balding (2010) is that the entropy measures how concentrated the posterior is, and thus how much information is contained within the sample. However, other measures of spread or informativeness could be used in the `crit` argument instead of `nn.ent`.

Since `mincrit` performs an exhaustive search of all subsets of z, which can potentially be computationally intensive, the function has been designed to allow the user to decrease the number of computations by restricting the search to particular subsets of interest. In particular, as with the `AS.select` function, the user can limit the search to subsets of a maximum size, using the `limit` argument. Internally, this calls the function `combmat` to produce subsets on which to perform the criterion. For example `combmat(4)` produces a matrix of all subsets of size 4, whereas the code `combmat(4,limit = 2)` computes a matrix of all 10 subsets of size 2 and below from 4 statistics, each row of the matrix indicating which of the 4 statistics are included in the subset:

```
      C1 C2 C3 C4
 [1,]  1  0  0  0
 [2,]  0  1  0  0
 [3,]  0  0  1  0
 [4,]  0  0  0  1
 [5,]  1  1  0  0
 [6,]  1  0  1  0
 [7,]  1  0  0  1
 [8,]  0  1  1  0
 [9,]  0  1  0  1
[10,]  0  0  1  1
```

In addition, the search can be limited by setting the argument `sumsubs` to a particular subset of initial summaries. This has the effect of only considering subsets containing those statistics. Alternatively, with the argument `do.only`, the user can specify certain summary subsets to consider. This can either be in matrix format like the output from `combmat`, or a vector of indices indicating rows of `combmat(k)` for which to compute the `crit` criterion.

To run the minimum criterion search algorithm, one could do:

```
> entchoice <- mincrit(obsstats, param, simstats, crit = nn.ent,
+                          do.only = 1:30)
```

This would only consider the first 30 subsets as specified in `combmat(ncol(obsstats))`.

The `mincrit` function returns a list object with the following components:

`critvals`: If `do.crit = TRUE`, a matrix representing the computed `crit` criterion values.

`best`: A matrix representing the best subset (which minimises `crit`).

`posssubs`: A matrix (or vector) of subsets considered by the search algorithm. This component reflects the choice of input `do.only`.

`sumsubs`: The index of the initial pool of statistics considered in the search. By default, this is set to `1:ncol(obsstats)`.

The best subset is judged to be the 20th subset in the search, $\{C_3, C_5\}$, as seen from the `best` component of the output:

```
> entchoice$best
   [,1] [,2]
20    3    5
```

Two stage procedure. As a refinement of the entropy-based summary selection procedure, Nunes and Balding (2010) propose running a second summary search based on the best subset found by minimum entropy. The closest *simulated* datasets to $x_{\rm obs}$ are identified using the summaries chosen in the first stage. The number of these close datasets is controlled by the argument `dsets`. The second stage selects a subset of summaries which minimises a measure of true posterior loss when ABC is performed on these datasets. This is done by comparing the ABC output to the true generating parameter values by some criterion. The default is calculating relative sum of squares error (RSSE). Since this second stage is effectively a search similar in form to that performed by `mincrit`, the functionality of `mincrit` is exploited by calling it internally within `stage2`. By default, the posterior loss minimisation is computed with the function `rsse`. The argument `init.best` specifies which subset to use as a basis to perform the second ABC analysis, e.g., the best subset chosen by the minimum entropy criterion. Other arguments to this function mimic those of `mincrit`.

An example call for this function is

```
> twostchoice <- stage2(obsstats, param, simstats, dsets = 25,
+                       init.best = 20, do.only = 1:30)
> twostchoice$best
   [,1] [,2]
21    3    6
```

The output object is the same as that of `mincrit`, with the exception that in addition, `stage2` also returns the `dsets` simulated datasets deemed closest to the observed data $z_{\rm obs}$.

Semi-automatic ABC

When the set of input statistics $z(x) = (z_1, z_2, \ldots, z_k)$ is large, it is computationally inefficient to search all possible subsets. Furthermore, good summary statistics for ABC may not be individual z_is but combinations e.g., their mean. *Semi-automatic ABC* (Fearnhead and Prangle, 2012) is a projection method which attempts to find linear combinations which are informative about θ by fitting a regression. This produces a low dimensional vector of summaries as there is one for each parameter, i.e., $\hat{\theta}_i(z) = \beta_{i0} + \sum_{j=1}^{k} \beta_{ij} z_j$ for $1 \leq i \leq p$ where p is the dimension of θ. The summaries are estimators of the conditional posterior parameter mean $\mathbb{E}(\theta|x)$. As theoretical support, Fearnhead and Prangle prove that ABC using $s(x) = \mathbb{E}(\theta|x)$ (i.e., perfect estimators) and $\varepsilon = 0$ would minimise a posterior loss function reflecting the quality of point estimators.

Linear regression is a crude tool to estimate $\mathbb{E}(\theta|x)$ so some further steps are proposed. These require some user input, which is why the method is referred to as *semi*-automatic. Firstly the set of input statistics z must be carefully chosen. For this method it should be composed of many potentially informative data features. These could include the raw data and various non-linear transformations for example. Secondly it is recommended to only fit the regression locally to the main posterior mass by using the following steps.

1. Perform an ABC pilot run using summary statistics chosen subjectively or using another method. Use this to determine the region of main posterior mass, referred to as the *training region*.
2. Simulate parameters θ^j_{train} from the prior truncated to the training region and corresponding datasets x^j_{train} for $1 \leq j \leq N$.
3. Fit regressions as detailed above for various choices of $z = z(x)$.
4. Choose the best fitting regression (e.g., using BIC) and run ABC using the corresponding summaries. For robustness it is necessary to truncate the prior to the training region; our experience is that without such truncation artefact posterior modes may appear outside the training region.

Note that in rejection-ABC the same simulations can be used for the pilot ABC, training and main ABC steps, if desired. Also, step 1 can be omitted and the entire parameter space used as the training region. This is simpler, but empirical evidence shows that in some situations the training step is crucial to good performance (Fearnhead and Prangle, 2012; Blum et al., 2013).

abctools provides two functions for semi-automatic ABC. To facilitate a quick analysis, `semiauto.abc` performs a simple complete analysis; this uses rejection-ABC, avoids selecting a training region (i.e., it uses the full parameter space instead), and uses a single prespecified choice of z. To allow the user to implement the full method, `saABC` implements step 3 only. We describe only the former here as the latter is a very straightforward function. The main arguments of `semiauto.abc` are:

`obs`: Input statistics corresponding to observed data. This is a matrix of dimension `ndatasets x k`. In fact only a subset $z'(x_{\text{obs}})$ need be supplied. The full vector $z(x_{\text{obs}})$ consists of deterministic transformations of these specified by `satr`.

`param`: Simulated parameters (drawn from a prior) which were used to generate simulated data under the model; a matrix of dimension `nsims x p`.

`sumstats`: Input statistics $z'(x)$ generated using the model with the parameters `param`; a matrix of dimension `nsims x k`.

`satr`: A `list` of *functions*, representing the vector of transformations to perform on the features `sumstats`, with which to estimate the relationship to the parameters θ. For more details, see the examples below.

Other arguments to the function are the same as `mincrit`; see the `saABC` documentation for more details.

To perform semi-automatic ABC using the vector of elementwise transformations (z', z'^2, z'^3, z'^4), one could use the function call:

```
> saabc <- semiauto.abc(obsstats, param, simstats,
+                       satr = list(function(x) {
+                         outer(x, Y = 1:4, "^")}))
```

Alternatively, the same transformations could be specified by setting `satr` to `list(function(x) cbind(x, x^2, x^3, x^4))`. This alternative way of choosing this argument uses a single `function` which outputs all four transformations as a vector.

The output from the `semiauto.abc` function is similar to that of `mincrit`, except that the output object also has a component `sainfo`, containing relevant choices of arguments pertaining to the ABC runs in steps 1 and 4 above. More specifically, the `sainfo` component is a list with information about the simulations used to perform each of the ABC runs, as well as the vector of transformations `satr`.

An example of `semiauto.abc` on the g-and-k dataset is as follows. The corresponding results and an analysis on another dataset are shown in Figure 2.

```
> mycon <- url("http://www.maths.lancs.ac.uk/~nunes/ABC/gkdata.rda")
> load(mycon)
> close(mycon)
> params <- gkdata[, 1:4]
> octiles <- gkdata[, 5:11]
> obs <- octiles[9, ]
> tfs <- list(function(x){cbind(x, x^2, x^3, x^4)})
> saabc <- semiauto.abc(obs = obs, param = params, sumstats = octiles,
+                       satr = tfs, overlap = TRUE, saprop = 1,
+                       abcprop = 1, tol = 0.001, method = "rejection",
+                       final.dens = TRUE)
> dens <- kde2d(saabc$post.sample[, 1], saabc$post.sample[, 2])
> filled.contour(dens, xlab = "A", ylab = "B")
```

An example on the `coal` data is as follows. Results are shown in Figure 3.

```
> data(coal)
> data(coalobs)
> coalparams <- coal[, 1:2]
> coaldata <- coal[, 3:9]
> coalobs <- coal[1, 3:9]
> mytf <- list(function(x){cbind(x, x^2, x^3, x^4)})
> saabc.coal <- semiauto.abc(obs = coalobs, param = coalparams,
+                            sumstats = coaldata, satr = mytf,
+                            tol = 0.001, overlap = TRUE, saprop = 1,
+                            abcprop = 1, method = "rejection",
+                            final.dens = TRUE)
> dens.coal <- kde2d(saabc.coal$post.sample[, 1],
```

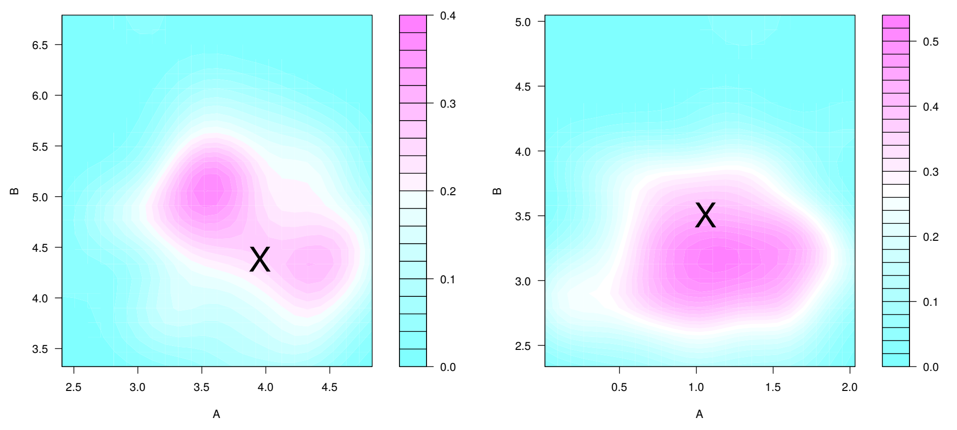

Figure 2: Joint posterior densities for two datasets for the (A, B) g-and-k distribution parameters, based on summary statistics chosen by semi-automatic ABC. The true parameter values are indicated by crosses.

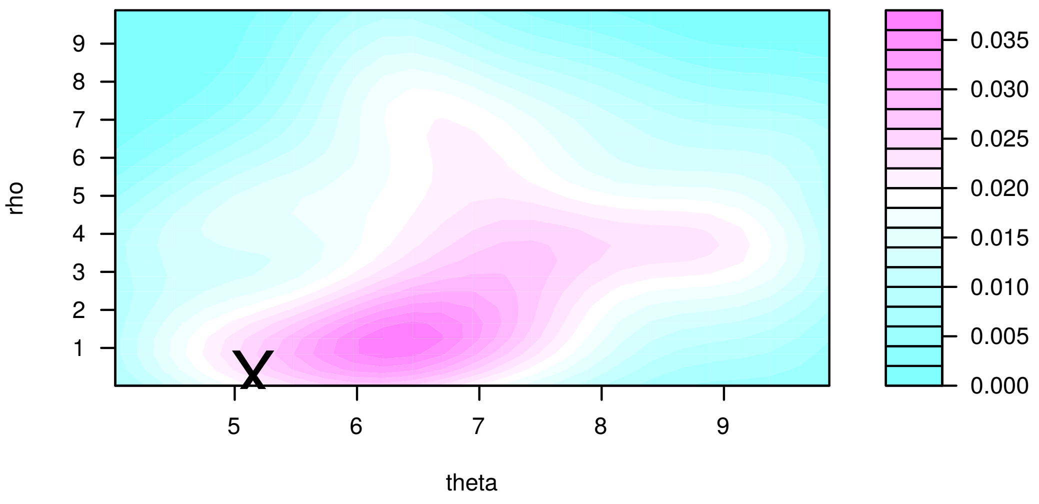

Figure 3: Joint posterior densities for the `coal` example, based on summary statistics chosen by semi-automatic ABC. The true parameter values are indicated by crosses.

```
+                     saabc.coal$post.sample[, 2])
> filled.contour(dens.coal, xlab = "theta", ylab = "rho")
```

The `selectsumm` convenience wrapper

The summary selection methods described in this section can be used with the individual functions as described above. Alternatively, the **abctools** package contains a convenient generic function `selectsumm`, with which any of the summary statistics choice algorithms can be performed. The argument `ssmethod` can be any of the functions described above, for example `mincrit`. Note that any other arguments to the `ssmethod` function can be passed to `selectsumm` easily. In particular, many of the summary selection routines have common optional arguments, for example

`obspar`: An optional matrix of true parameters corresponding to the observed summaries `obs`. This is useful if the function is used to test summary selection techniques on fake observed data (for which you know the generating parameters).

`abcmethod`: A function which performs an ABC algorithm, for example the `abc` function from the **abc** R package. Other user-defined functions can also be supplied; see below for more details. By default, the `ssmethod` function uses the `abc` rejection-ABC algorithm, with a tolerance of `tol = 0.01`.

`limit`: An (optional) integer value indicating whether to limit the search to subsets of a particular maximum size. For example, `limit = 3` would only consider potential subsets of statistics s_A with $|s_A| = 3$, see the subset selection section for more details.

`do.err`: A logical variable indicating whether the simulation error should be computed to assess the performance of the selection algorithm. This is only relevant if `obspar` is supplied.

`final.dens`: A logical variable. If `final.dens = TRUE`, then the final approximate posterior sample is returned, resulting from the ABC algorithm (`abcmethod`) using the final subset of summaries s_A.

`errfn`: A function used to compute the simulation error between the posterior sample and the generating parameter values `obspar`. An example of such a function included in the **abctools** package is the relative sum of squares error (RSSE), computed using the function `rsse`.

Note that the `selectsumm` function can perform summary selection for any number of observed summary vectors; the function implements the `ssmethod` on each row of the `obsstats` argument. Examples of the `selectsumm` function call are

```
> ASchoice <- selectsumm(obsstats, param, simstats, ssmethod = AS.select)
```

or

```
> mycon <- url("http://www.maths.lancs.ac.uk/~nunes/ABC/gkdata.rda")
> load(mycon)
> close(mycon)
> param <- gkdata[, 1:2]                # A and B parameters
> simstats <- gkdata[, 5:11]
> obsstats <- gkdata[9:10, 5:11]        # treated as real data
> entchoicegk <- selectsumm(obsstats, param, simstats, ssmethod = mincrit,
+                          crit = nn.ent, limit = 3, final.dens = TRUE,
+                          do.err = TRUE, obspar = gkdata[9:10, 1:2])
> entchoicegk$best
   S1  S2  S3  S4  S5  S6  S7
23  0   0   0   1   1   0   0
23  0   0   0   1   1   0   0
```

If `do.err = TRUE`, then the inference error (as compared with the truth in `obspar`) is computed using the `errfn` function and is also returned in the `err` component of the function output. In addition, if `final.dens = TRUE` the output list element `post.sample` will contain the approximate posterior sample from the ABC inference corresponding to using s_A in the `abcmethod` ABC inference function. For example, for the `entchoicegk` object, the approximate posterior sample corresponds to the algorithm `abcmethod` using the subset (of size ≤ 3) with the lowest entropy. The resulting bivariate posterior density can then be seen by using the command `kde2d` from package **MASS** (Venables and Ripley, 2002):

```
> dens1 <- kde2d(entchoicegk$post.sample[, 1, 1],
+                entchoicegk$post.sample[, 2, 1])
> dens2 <- kde2d(entchoicegk$post.sample[, 1, 2],
+                entchoicegk$post.sample[, 2, 2])
> filled.contour(dens1, xlab = "A", ylab = "B")
> filled.contour(dens2, xlab = "A", ylab = "B")
```

The resulting posterior densities are shown in Figure 4.

Any other arguments to be passed to the function specified by the `abcmethod` argument can also be included. For more details on the optional arguments for the `abc` function see Csillèry et al. (2012).

Using other ABC algorithms with abctools

The flexibility of the **abctools** package can be exploited by using user-defined ABC algorithm implementations through the `abcmethod` argument to all of the ABC summary choice methods, namely `AS.select`, `mincrit`, `stage2` and `semiauto.abc`, or the convenience wrapper `selectsumm`, described above. The only constraint on the user's code for the ABC method is that it must return an object with a component named either `adj.values` or `unadj.values` containing the approximate posterior sample, to (minimally) mimic a return object of class `"abc"`. For example, if one had written a function `likefreemcmc` to perform likelihood-free Markov chain Monte Carlo, one could use this in combination with a minimal criterion computed on the resulting (MCMC) posterior samples using the code:

```
> mcmcabc <- mincrit(obsstats, param, simstats, abcmethod = likefreemcmc)
```

To use **abctools** within ABC inference methods implemented by generic software, simply supply an appropriate R wrapper function to the `abcmethod` argument.

User-defined ABC summary selection methods can be accommodated with the **abctools** package. A new *projection* method, `projABC` say, could be implemented using the wrapper `selectsumm` as follows:

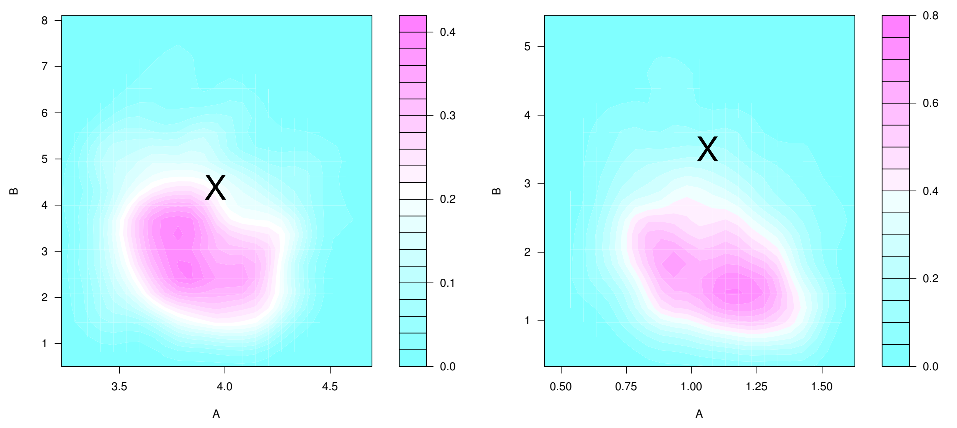

Figure 4: Joint posterior densities for two datasets for the (A, B) g-and-k distribution parameters, based on the $\{s_4, s_5\}$ statistics, as chosen by the minimum entropy subset selection method. The true parameter values are indicated by crosses.

```
> projchoice <- selectsumm(obsstats, param, simstats, ssmethod = projABC)
```

For the implementation to work, the summary choice function must have arguments named `obsstats`, `param`, `simstats` for the observed data, simulated parameters and simulated summaries respectively, as well as the logical argument `final.dens` indicating whether the approximate posterior sample is to be returned. Optional arguments could also be passed to `projABC` through the `selectsumm` wrapper.

Coverage

Theory

The **abctools** package can also test the accuracy of an ABC analysis, in particular to help choose the ε tuning parameter. This is done by testing whether it satisfies the *coverage property* (Prangle et al., 2014). As a simple example, consider (exact) Bayesian inference for the scalar parameter θ given data x. A standard summary of this is an α% credible interval: an interval I such that $\Pr(\theta \in I|x) = \alpha/100$. Suppose a dataset is simulated from a parameter value θ_0 drawn from the prior and an α% credible interval is calculated. It is easy to show that the probability the interval contains θ_0 is $\alpha/100$. When this is true for all α, an inference method is said to satisfy the coverage property. This is illustrated by Figure 5. Note that the probability in question relates to a *random choice of* θ_0. The stricter requirement of frequentist coverage requires a similar condition holds *for every* θ_0.

Monahan and Boos (1992) and Cook et al. (2006) showed that an equivalent condition to the coverage property is that the distribution of p_0, the posterior quantile of θ_0, must be $U(0,1)$. This property is much easier to test numerically, as shown in Figure 6. Prangle et al. (2014) discuss how such a test can be implemented efficiently in a rejection-ABC context. This involves performing ABC analyses under many data sets simulated from known θ_0 values. A manageable computational cost is achieved by reusing the same ABC simulations in each analysis and exploiting multicore processing. Prangle et al. (2014) also show that when θ_0 values are drawn from the prior, the coverage property is a necessary condition for an inference procedure to give the correct posterior but not a sufficient condition: the coverage property also holds for an inference procedure that always returns the prior distribution (see Figure 5). Recommendations are given for how this problem can be avoided, involving drawing θ_0 values from a non-prior distribution, but are not discussed here for reasons of brevity.

Prangle et al. (2014) also discuss testing the coverage property in ABC model choice analyses. The idea is to test that amongst analyses giving model $\mathcal{M}$ weight of roughly α, the proportion of being truly from model $\mathcal{M}$ is close to α. Several test statistics are proposed which can be calculated with **abctools**.

Package usage

The following code illustrates a typical analysis using the `cov.pi` (parameter inference) and `cov.mc` (model choice) functions. For the choice of tolerance ε, the user must supply simulated parameters, summary statistics and, for model choice, model indicators. Figure 7 shows typical output (n.b. some code to improve the appearance of this figure has been omitted.)

```
> library(abctools); library(ggplot2)
> data(human)
```

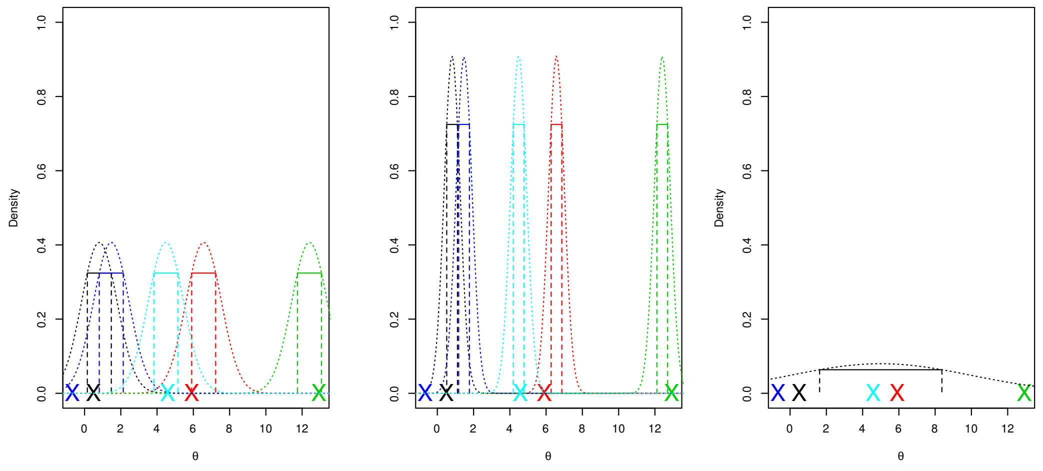

Figure 5: Illustration of the coverage property. The crosses represent simulated θ_0 values. The dotted curves on the left graph show the posterior densities based on noisy data about θ. On the middle graph they show approximate posterior densities which are over-precise, and the right-hand graph shows the prior density. All graphs have the same θ_0 and data values. The horizontal lines show 50% credible intervals. In the case shown on the left roughly half of these will contain the corresponding θ_0 value, which is consistent with the coverage property. For the middle graph case the proportion is generally smaller, illustrating that the coverage property does not hold. The right-hand graph shows that the prior credible interval also contains roughly half the θ_0 values, illustrating that coverage also holds here.

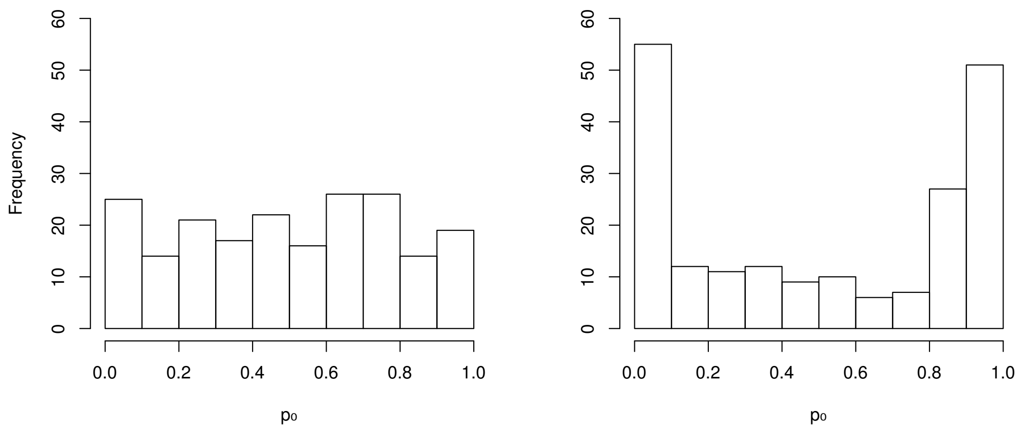

Figure 6: Illustration of testing the coverage property via the distribution of p_0 values, extending the example of Figure 5. The left-hand histogram shows the case where 200 p_0 values are calculated from the posterior distribution. On the right an over-precise estimate of the posterior is used instead. Clearly the left-hand histogram is consistent with $p_0 \sim U(0,1)$ approximately and the right-hand one is not. This is confirmed by using the Kolmogorov-Smirnov test, which gives p-values of 0.82 (left) and 10^{-7} (right).

```
> ## Summary statistics for bottleneck model:
> stat.italy.sim <- subset(stat.3pops.sim, subset = (models == "bott"))
> ## Interesting epsilon values:
> myeps <- exp(seq(log(0.5), log(10), length.out = 15))
> set.seed(1)
> mytestsets <- sample(1:nrow(stat.italy.sim), 200)
> covout.pi <- cov.pi(param = par.italy.sim, sumstat = stat.italy.sim,
+                     testsets = mytestsets, eps = myeps,
+                     multicore = TRUE, diagnostics = c("KS", "CGR"),
+                     cores = 4)
> qplot(x = eps, y = pvalue, colour = test,
+       data = subset(cabc.out$diag, parameter == "Ne"), log = "y")
> mytestsets <- sample(nrow(stat.3pops.sim), 200)
> covout.mc <- cov.mc(index = models, sumstat = stat.3pops.sim,
```

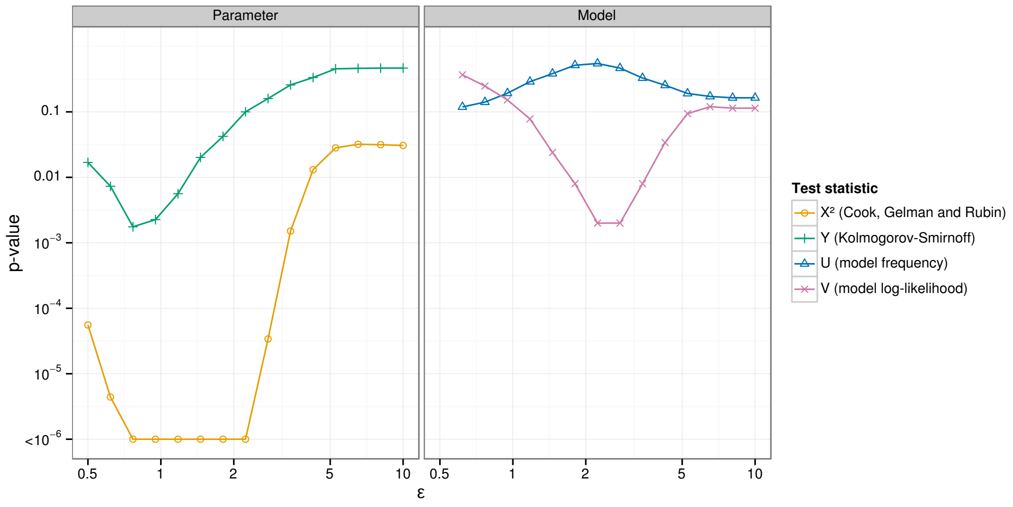

Figure 7: p-values testing coverage in the `human` dataset example. The left-hand graph is for the N_e parameter in the `bottleneck` model. The right-hand graph is for model choice considering the adequacy of the `bottleneck` model predictions.

```
+                       testsets = mytestsets, eps = myeps,
+                       diagnostics = c("freq", "loglik.binary"),
+                       multicore = TRUE, cores = 4)
> qplot(x = eps, y = pvalue, colour = test, data = covout.mc$diag,
+       log = "y")
```

The code analyses the `human` dataset supplied in the **abc** package (Csilléry et al., 2012), which contains simulated parameter values and summary statistics for a population genetic model. The `cov.pi` function estimates 200 p_0 values for each parameter. To do this, 200 of the simulated datasets are randomly sampled to be used as *pseudo-observed data* in leave-one-out style ABC analyses. The p-values of various diagnostic test statistics are returned in the `diag` component of the output. The left panel of Figure 7 plots p-values of uniformity tests – Kolmogorov-Smirnov and that of Cook et al. (2006) – as ε varies for one particular parameter. These show typical behaviour; coverage is supported for large ε when the ABC output is approximately drawn from the prior and also for ε small enough that ABC output is approximately drawn from the posterior. The right panel shows p-values for tests of whether output for the `bottleneck` model satisfies coverage. Again, coverage holds for large and small ε, but there is disagreement in between.

Prangle et al. (2014) argue that p-values of test statistics only investigate certain aspects of coverage. A fuller investigation of interesting ε values, for example where test statistics disagree, can be found by diagnostic plots. For parameter inference histograms of underlying p_0 values are recommended, and for model choice plots of estimated against observed model probabilities, after some aggregation. This information is returned in the `raw` component of the output and can be plotted as follows, giving Figure 8. The `mc.ci` command is part of **abctools**.

```
> par(mfrow = c(1, 2))
> ## nb myeps[8] is 2.2
> hist(subset(covout.pi$raw, eps == myeps[8])$Ne, xlab = "p0",
+      main = "Parameter inference")
> mc.ci(covout.mc$raw, eps = myeps[8], modname = "bott",
+       modtrue = models, main = "Model choice")
```

The left-hand side of Figure 8 shows that for $\varepsilon = 2.2$, coverage clearly does not hold for the parameter of interest. The right-hand side shows no evidence to reject coverage for the `bottleneck` model.

Summary

This article has described the R package **abctools**. This implements several techniques for tuning approximate Bayesian inference algorithms. In particular, the package contains summary statistic selection routines for the approximate sufficiency method of Joyce and Marjoram (2008); the entropy minimisation and two-stage error algorithm proposed by Nunes and Balding (2010); and the regression method of Fearnhead and Prangle (2012). It also contains methods to choose the acceptance threshold ε by assessing the coverage property of Prangle et al. (2014).

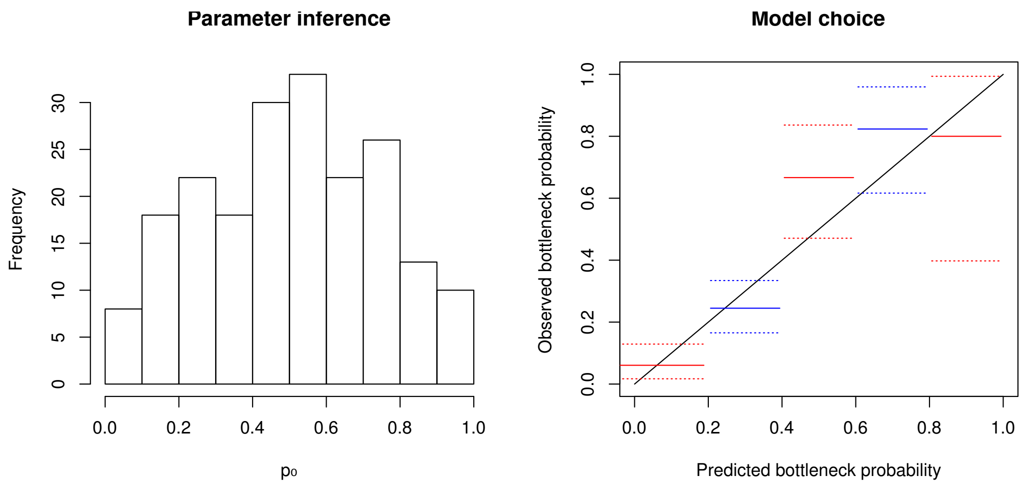

Figure 8: Detailed coverage diagnostic plots for the `human` dataset example with $\varepsilon = 2.2$.

Acknowledgements

The authors would like to thank Scott Sisson and Michael Blum for some helpful suggestions when preparing the **abctools** R package accompanying this manuscript.

Bibliography

C. N. K. Anderson, U. Ramakrishnan, Y. L. Chan, and E. A. Hadly. Serial SimCoal: A population genetics model for data from multiple populations and points in time. *Bioinformatics*, 21(8):1733–1734, 2005. [p3]

S. Barthelmé and N. Chopin. Expectation propagation for likelihood-free inference. *Journal of the American Statistical Association*, 109(505):315–333, 2014. [p3]

M. A. Beaumont. Approximate Bayesian computation in evolution and ecology. *Annual Review of Ecology, Evolution and Systematics*, 41:379–406, 2010. [p1, 2]

M. A. Beaumont, W. Zhang, and D. J. Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002. [p2]

M. A. Beaumont, J.-M. Marin, J.-M. Cornuet, and C. P. Robert. Adaptive approximate Bayesian computation. *Biometrika*, 96(4):983–990, 2009. [p2]

M. G. B. Blum and O. François. Non-linear regression models for approximate Bayesian computation. *Statistics and Computing*, 20(1):63–73, 2010. [p2, 3, 4]

M. G. B. Blum, M. A. Nunes, D. Prangle, and S. A. Sisson. A comparative review of dimension reduction methods in approximate Bayesian computation. *Statistical Science*, 28(2):189–208, 2013. [p2, 3, 4, 6, 8]

E. Brooks-Pollock, G. O. Roberts, and M. J. Keeling. A dynamic model of bovine tuberculosis spread and control in Great Britain. *Nature*, 511(7508):228–231, 2014. [p1]

E. Cameron and A. N. Pettitt. Approximate Bayesian computation for astronomical model analysis: A case study in galaxy demographics and morphological transformation at high redshift. *Monthly Notices of the Royal Astronomical Society*, 425(1):44–65, 2012. [p1]

S. R. Cook, A. Gelman, and D. B. Rubin. Validation of software for Bayesian models using posterior quantiles. *Journal of Computational and Graphical Statistics*, 15(3):675–692, 2006. [p11, 13]

J.-M. Cornuet, F. Santos, M. A. Beaumont, C. P. Robert, J.-M. Marin, D. J. Balding, T. Guillemaud, and A. Estoup. Inferring population history with DIYABC: A user-friendly approach to approximate Bayesian computation. *Bioinformatics*, 24(23):2713–2719, 2008. [p3]

J.-M. Cornuet, V. Ravigné, and A. Estoup. Inference on population history and model checking using DNA sequence and microsatellite data with the software DIYABC (v1. 0). *BMC Bioinformatics*, 11(401), 2010. [p3]

J.-M. Cornuet, P. Pudlo, J. Veyssier, A. Dehne-Garcia, M. Gautier, R. Leblois, J.-M. Marin, and A. Estoup. DIYABC v2.0: A software to make approximate Bayesian computation inferences about population history using single nucleotide polymorphism, DNA sequence and microsatellite data. *Bioinformatics*, 30(8):1187–1189, 2014. [p3]

K. Csilléry, M. G. B. Blum, O. E. Gaggiotti, and O. François. Approximate Bayesian computation (ABC) in practice. *Trends in Ecology & Evolution*, 25(7):410–418, 2010. [p1, 2]

K. Csilléry, O. François, and M. G. B. Blum. abc: An R package for approximate Bayesian computation (ABC). *Methods in Ecology and Evolution*, 2012. [p1, 2, 3, 10, 13]

P. Del Moral, A. Doucet, and A. Jasra. An adaptive sequential Monte Carlo method for approximate Bayesian computation. *Statistics and Computing*, 22(5):1009–1020, 2012. [p2, 3]

C. C. Drovandi and A. N. Pettitt. Likelihood-free Bayesian estimation of multivariate quantile distributions. *Computational Statistics and Data Analysis*, 55(9):2541–2556, 2011. [p3, 4]

A. Estoup, E. Lombaert, J.-M. Marin, T. Guillemaud, P. Pudlo, C. P. Robert, and J.-M. Cornuet. Estimation of demo-genetic model probabilities with approximate Bayesian computation using linear discriminant analysis on summary statistics. *Molecular Ecology Resources*, 12(5):846–855, 2012. [p3]

N. J. R. Fagundes, N. Ray, M. A. Beaumont, S. Neuenschwander, F. M. Salzano, S. L. Bonatto, and L. Excoffier. Statistical evaluation of alternative models of human evolution. *Proceedings of the National Academy of Sciences of the United States of America*, 104(45):17614–17619, 2007. [p1]

P. Fearnhead and D. Prangle. Constructing summary statistics for approximate Bayesian computation: Semi-automatic ABC (with discussion). *Journal of the Royal Statistical Society: Series B*, 74(3):419–474, 2012. [p4, 7, 8, 13]

M. J. Hickerson, E. Stahl, and N. Takebayashi. msBayes: Pipeline for testing comparative phylogeographic histories using hierarchical approximate Bayesian computation. *BMC Bioinformatics*, 8(268), 2007. [p3]

W. Huang, N. Takebayashi, Y. Qi, and M. J. Hickerson. MTML-msBayes: Approximate Bayesian comparative phylogeographic inference from multiple taxa and multiple loci with rate heterogeneity. *BMC Bioinformatics*, 12(1), 2011. [p3]

F. Jabot, T. Faure, and N. Dumoulin. EasyABC: Performing efficient approximate Bayesian computation sampling schemes using R. *Methods in Ecology and Evolution*, 4(7):684–687, 2013. [p3]

M. J. Jobin and J. L. Mountain. REJECTOR: Software for population history inference from genetic data via a rejection algorithm. *Bioinformatics*, 24(24):2936–2937, 2008. [p3]

P. Joyce and P. Marjoram. Approximately sufficient statistics and Bayesian computation. *Statistical Applications in Genetics and Molecular Biology*, 7(1), 2008. Article 26. [p3, 4, 5, 13]

M. Lenormand, F. Jabot, and G. Deffuant. Adaptive approximate Bayesian computation for complex models. *Computational Statistics*, 28(6):2777–2796, 2013. [p3]

J. Liepe, C. Barnes, E. Cule, K. Erguler, P. Kirk, T. Toni, and M. P. H. Stumpf. ABC-SysBio – approximate Bayesian computation in python with GPU support. *Bioinformatics*, 26(14):1797–1799, 2010. [p3]

J. S. Lopes, D. Balding, and M. A. Beaumont. PopABC: A program to infer historical demographic parameters. *Bioinformatics*, 25(20):2747–2749, 2009. [p3]

F. Luciani, S. A. Sisson, H. Jiang, A. R. Francis, and M. M. Tanaka. The epidemiological fitness cost of drug resistance in Mycobacterium tuberculosis. *Proceedings of the National Academy of Sciences of the United States of America*, 106(34):14711–14715, 2009. [p1]

J.-M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder. Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012. [p2]

P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré. Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences of the United States of America*, 100(26):15324–15328, 2003. [p2]

J. F. Monahan and D. D. Boos. Proper likelihoods for Bayesian analysis. *Biometrika*, 79(2):271–278, 1992. [p11]

M. Nordborg. Coalescent theory. In D. J. Balding, M. J. Bishop, and C. Cannings, editors, *Handbook of Statistical Genetics*, pages 179–208. Wiley, Chichester, third edition, 2007. [p3]

M. A. Nunes and D. J. Balding. On optimal selection of summary statistics for approximate Bayesian computation. *Statistical Applications in Genetics and Molecular Biology*, 9(1), 2010. [p3, 4, 6, 7, 13]

U. Picchini. abc-sde: Approximate Bayesian computation for stochastic differential equations, 2013. URL `http://sourceforge.net/projects/abc-sde/`. [p3]

D. Prangle, M. G. B. Blum, G. Popovic, and S. A. Sisson. Diagnostic tools for approximate Bayesian computation using the coverage property. *Australia and New Zealand Journal of Statistics*, 56(4):309–329, 2014. [p1, 2, 11, 13]

O. Ratmann, O. Jørgensen, T. Hinkley, M. P. H. Stumpf, S. Richardson, and C. Wiuf. Using likelihood free inference to compare evolutionary dynamics of the protein networks of H. pylori and P. falciparum. *PLoS Computational Biology*, 3(11):2266–2278, 2007. [p1]

G. D. Rayner and H. L. MacGillivray. Numerical maximum likelihood estimation for the g-and-k and generalized g-and-h distributions. *Statistics and Computing*, 12(1):57–75, 2002. [p4]

H. Singh, V. Misra, N.and Hnizdo, A. Fedorowicz, and E. Demchuk. Nearest neighbor estimates of entropy. *American Journal of Mathematical and Management Sciences*, 23(3–4):301–321, 2003. [p6]

S. A. Sisson and Y. Fan. Likelihood-free Markov chain Monte Carlo. In S. P. Brooks, A. Gelman, G. Jones, and X.-L. Meng, editors, *Handbook of Markov Chain Monte Carlo*, pages 319–341. Chapman and Hall/CRC Press, 2011. [p2]

S. A. Sisson, Y. Fan, and M. Tanaka. Sequential Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences of the United States of America*, 104:1760–1765, 2007. Errata (2009), 106, 16889. [p2]

D. A. Tallmon, A. Koyuk, G. Luikart, and M. A. Beaumont. onesamp: A program to estimate effective population size using approximate Bayesian computation. *Molecular Ecology Resources*, 8(2):299–301, 2008. [p3]

K. R. Thornton. Automating approximate Bayesian computation by local linear regression. *BMC Genetics*, 10 (35), 2009. [p3]

T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P. H. Stumpf. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society and its Interface*, 6(31):187–202, 2009. [p1, 2]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL `http://www.stats.ox.ac.uk/pub/MASS4`. [p10]

D. Wegmann, C. Leuenberger, and L. Excoffier. Efficient approximate Bayesian computation coupled with Markov chain Monte Carlo without likelihood. *Genetics*, 182(4):1207–1218, 2009. [p3, 4]

D. Wegmann, C. Leuenberger, S. Neuenschwander, and L. Excoffier. ABCtoolbox: A versatile toolkit for approximate Bayesian computations. *BMC Bioinformatics*, 11(116), 2010. [p3]

A. Weyant, C. Schafer, and W. M. Wood-Vasey. Likelihood-free cosmological inference with type Ia supernovae: Approximate Bayesian computation for a complete treatment of uncertainty. *The Astrophysical Journal*, 764 (2):116, 2013. [p1]

Matthew A. Nunes
Lancaster University
Lancaster
UK
`m.nunes@lancaster.ac.uk`

Dennis Prangle
University of Reading
Reading
UK
`d.b.prangle@reading.ac.uk`