Approximate Bayesian Computation (ABC) in R: A Vignette

K Csilléry, O François, MGB Blum

`abc` version 1.4 , 2011-09-03

Contents

[1]This document is included as a vignette (a LaTeX document created using the R function `Sweave`) of the package `abc`. It is automatically dowloaded together with the package and can be accessed through R typing `vignette("abc")`.

Summary

An implementation of Approximate Bayesian Computation (ABC) methods in the R language is available in the package `abc`. The aim of this vignette is to provide an extended overview of the capabilities of the package, with a detailed example of the analysis of real data. Some general information on installation procedures and object oriented programming in R are given in Section *Introduction*. The ABC methodology is briefly illustrated in Section *A brief introduction to ABC*. A detailed example of an ABC analysis, where we aim to estimate the ancestral human population size is provided in Section *A walk-though example: human demographic history*. Users already familiar with the R and ABC methods may skip the first two Sections and start with the example.

Introdunction

Installation

R is a open source software project and can be freely downloaded from the CRAN website. There are already several resources for an introduction to R . The CRAN website link to contributed documentation also offers excellent introductions in several languages. Once R is running the installation of additional packages is quite straightward. To install the `abc` package from R simply type:

```
> install.packages("abc")
```

Once the `abc` package is installed, it needs to be made accessible to the current R session by the command:

```
> library(abc)
```

For online help facilities or the details of a particular command (such as the function `abc`) you can type:

```
> help(package = "abc")
> help(abc)
```

The first command gives a brief summary of the available commands in the package, the second give more detailed information about a specific command. R help files can also be used to illustrate how commands are executed, so they can be pasted into an R session, or run as a whole with the following command:

```
> example(abc)
```

Methods and classes in R

This is only a brief reminder that expressions in R manipulate objects, which may be data (a scalar or a matrix for example), but objects may also be functions, or more complex collections of objects. All objects have a class, which enables functions to act on them appropriately. Thus, for example, when the function `summary` is applied on an object of class `"abc"` (i.e. an object that had been generated by the function `abc`), it would act differently than on a simple matrix. In fact, the function `summary` on an `"abc"` object calls the function `summary.abc`. Likewise, the `plot`, `hist` and `print` functions will recognize the classes of objects generated by the various functions of the `abc` package and act accordingly.

A brief introduction to ABC

This section contains a short introduction to the ABC methodology. Recall that the main steps of an ABC analysis follow the general scheme of any Bayesian analysis: formulating a model, fitting the model to data (parameter estimation), and improving the model by checking its fit (posterior predictive checks) and comparing it to other models (Csilléry et al., 2010; Gelman et al., 2003). In the following sections, we detail each of these steps and highlight the appropriate functions of the `abc` package.

Parameter inference

Suppose that we want to compute the posterior probability distribution of a univariate or multivariate parameter, θ. A parameter value θ_i, is sampled from its prior distribution to simulate a dataset y_i, for $i = 1, \ldots, n$ where n is the number of simulations. A set of summary statistics $S(y_i)$ is computed from the simulated data and compared to the summary statistics obtained from the actual data $S(y_0)$ using a distance measure d. We consider the Euclidean distance for d, where each summary statistic is standardized by a robust estimate of the standard deviation (the median absolute deviation). If $d(S(y_i), S(y_0))$ (i.e. the distance between $S(y_i)$ and $S(y_0)$) is less than a given threshold, the parameter value θ_i is accepted. In order to set a threshold above which simulations are rejected, the user has to provide the tolerance rate, which is defined as the percentage of accepted simulation. The accepted θ_i's form a sample from an approximation of the posterior distribution. The estimation of the posterior distribution can be improved by the use of regression techniques (see below).

The `abc` package implements three ABC algorithms for constructing the posterior distribution from the accepted θ_i's: a rejection method, and regression-based correction methods that use either local linear regression (Beaumont et al., 2002) or neural networks (Blum and François, 2010). When the rejection method is selected, the accepted θ_i's are considered as a sample from the posterior distribution (Pritchard et al., 1999). The two regression methods implement an additional step to correct for the imperfect match between the accepted, $S(y_i)$, and observed summary statistics, $S(y_0)$, using the following regression equation in the vicinity of $S(y_0)$

$$\theta_i = m(S(y_i)) + \epsilon_i,$$

where m is the regression function, and the ϵ_i's are centered random variables with a common variance. Simulations that closely match $S(y_0)$ are given more weight by assigning to each simulation $(\theta_i, S(y_i))$ the weight $K[d(S(y_i), S(y_0))]$, where K is a statistical kernel. The local linear model assumes a linear function for m (`"loclinear"` method in the `abc` function), while neural networks (`"neuralnet"` method) account for the non-linearity of m and allow users to reduce the dimension of the set of summary statistics. Once the regression is performed, a weighted sample from the posterior distribution is obtained by correcting the θ_i's as follows,

$$\theta_i^* = \hat{m}(S(y_0)) + \hat{\epsilon_i},$$

where $\hat{m}(\cdot)$ is the estimated conditional mean and the $\hat{\epsilon_i}$'s are the empirical residuals of the regression (Beaumont et al., 2002). Additionally, a correction for heteroscedasticity is applied (by default)

$$\theta_i^* = \hat{m}(S(y_0)) + \frac{\hat{\sigma}(S(y_0))}{\hat{\sigma}(S(y_i))}\hat{\epsilon_i}$$

where $\hat{\sigma}(\cdot)$ is the estimated conditional standard deviation (Blum and François, 2010). When using the `"loclinear"` method, a warning about the collinearity of the design matrix of the regression might be issued. In that situation, we recommend to rather use the related `"ridge"` method that performs local-linear ridge regression and deals with the collinearity issue.

Finally, we note that alternative algorithms exist that sample from an updated distribution that is closer in shape to the posterior than to the prior (Beaumont et al., 2009; Marjoram et al., 2003; Wegmann et al., 2010). However, we do not implement these methods in R `abc` because they require the repeated use of the simulation software.

Cross-validation

R `abc` implements a leave-one-out cross-validation to evaluate the accuracy of parameter estimates and the robustness of the estimates to the tolerance rate. To perform cross-validation, the i^{th} simulation is randomly selected as a validation simulation, its summary statistic(s) $S(y_i)$ are used as pseudo-observed summary statistics, and its parameters are estimated with the function `abc` using all simulations except the i^{th} simulation. Ideally, the process is repeated n times, where n is the number of simulations (so-called n-fold cross-validation). However, performing an n-fold cross-validation could be very time consuming, so the cross-validation is often performed for a subset of a few 100 randomly selected simulations. The prediction error is calculated as

$$E_{\text{pred}} = \frac{\sum_i (\tilde{\theta}_i - \theta_i)^2}{Var(\theta_i)},$$

where θ_i is the true parameter value of the i^{th} simulated data set and $\tilde{\theta}_i$ is the estimated parameter value (the posterior median, mean or mode).

Model selection

Model selection is part of any Bayesian analysis, and can be performed in an ABC framework. The aim is generally to estimate the posterior probability of a model M as $Pr(M|S(y_0))$. Three different methods are implemented in the `abc` package. With the rejection method, the posterior probability of a given model is approximated by the proportion of accepted simulations given this model. The two other methods are based on multinomial logistic regression or neural networks. In these two approaches, the model indicator is treated as the response variable of a polychotomous regression, where the summary statistics are the independent variables (Beaumont, 2008; Fagundes et al., 2007). Using neural networks can be efficient when highly dimensional statistics are used. Any of these methods are valid when the different models to be compared are, a priori, equally likely, and the same number of simulations are performed under each model.

A further function, `expected.deviance`, is implemented to guide the model selection procedure. The function computes an approximate expected deviance from the posterior predictive distribution. Thus, in order to use the function, users have to re-use the simulation tool and to simulate data from the posterior parameter values. The method is particularly advantageous when it is used with one of the regression methods. Further details on the method can be found in (François and Laval, 2011) and fully worked out examples are provided in the package manual.

Model misclassification

A cross-validation tool is available for model selection as well. The objective is to evaluate if model selection with ABC is able to distinguish between the proposed models by making use of the existing simulations. The summary statistics from one of the simulations are considered as pseudo-observed summary statistics and classified using all the remaining simulations. Then, one expects that a large posterior probability should be assigned to the model that generated the presudo-observed summary statistics. Two versions of the cross-validation are implemented. The first version is a "hard" model

classification. We consider a given simulation as the pseudo-observed data and assign it to the model with the highest posterior model probability. This procedure is repeated for a given number of simulations for each model. The results are summarized in a so-called *confusion matrix* (Hastie et al., 2009). Each row of the confusion matrix represents the number of simulations under a true model, while each column represents the number of simulations under a model assigned by `postpr`. If all simulations had been correctly classified, only the diagonal elements of the matrix are non-zero. The second version is called "soft" classification. Here we do not assign a simulation to the model with the highest posterior probability, but average the posterior probabilities over many simulations for a given model. This procedure is again summarized as a matrix, which is similar to the confusion matrix. However, the elements of the matrix do not give model counts, but the average posterior probabilities across simulations for a given model.

A walk-though example: human demographic history

Getting started

In following sections we show how the `abc` package can be used for the inference and model comparison steps, and we indicate how generic tools of the R package can be used for model checking. The package is a so-called "post-processing tool", so the simulations and the calculation of the summary statistics have to be performed by the user's preferred software. Alternatively, simulation software might be called in an R session, which opens up the possibility for a highly interactive ABC analysis. For example, for coalescent models, users might want to use one of the many existing software for simulating genetic data such as `ms` (Hudson, 2002) or `simcoal2` (Laval and Excoffier, 2004). Note that the package's data set `human`, for example, has been generated using `ms`. The exact code can be found in `inst/doc/runms.R` and may be re-run or modified by interested users. Since the relase of the `phyclust` package, a function called `ms` is also available, which allows users to call `ms` directly from an R session. The calculation of summary statistics for could either be done using our own R code or, for `ms` outputs, one could use the software `msABC` (Pavlidis et al., 2010) (`http://www.bio.lmu.de/~pavlidis/home/?Software:msABC`).

In order to use the package certain R objects always have to be prepared. These are a vector of the observed summary statistics, a matrix of the simulated summary statistics, where each row corresponds to a simulation and each column corresponds to a summary statistic, and a matrix of the simulated parameter values, where each row corresponds to a simulation and each column corresponds to a parameter. If users want to perform model selection, an additional vector with the model indices has to be ready.

The `abc` package contains two examples: `musigma2` and `human`. The `musigma2` data is very simple, so users might prefer to try this first. The example is motivated by Anderson, Edgar's iris data, so the observed statistics are the mean and variance of the sepal of *Iris setosa*, estimated from part of the `iris` data. The aim is to estimate two parameters: μ and σ^2. The advantage of this simple example is that the true posterior distributions are known, so the accuracy of parameter inference can be easily checked. The data set can be loaded using the following code:

```
> library(abc)
> data(musigma2)
```

Tha data set contains five R objects. The manual page of these R objects can be accessed by typing the name of one of the five objects, such as `stat.obs`, which contains the observed summary statistics (`?stat.obs`). At the end of the manual page of the function `abc`, several examples can be found using the `musigma2` data set, so we leave the user to discover this simple example on their own. This example can be accessed by typing `?abc`.

The human data

The `human` data set is more realistic. Our aim is to estimate the human ancestral population size and to discriminate between different demographic models. We discuss this example in detail.

Background

Several population genetic studies have found evidence that African human populations have been expanding, while human populations outside of Africa went through a out-of-Africa population bottleneck and then expanded. We re-examined this problem by re-analyzing published data of 50 unlinked autosomal non-coding regions from a Hausa (Africa), a Chinese (Asia), and an Italian (Europe) population (Voight et al., 2005). Data is summarized using three summary statistics: the average nucleotide diversity ($\bar{\pi}$), and the mean and the variance of Tajima's D (Table 1 in Voight et al. (2005)). Both Tajima's D and $\bar{\pi}$ have been classically used to detect historical changes in population size. The observed summary statistics (our data!) can be accessed with the following code:

```
> data(human)
> stat.voight
```

```
             pi TajD.m TajD.v
hausa   0.00110  -0.20   0.55
italian 0.00085   0.28   1.19
chinese 0.00079   0.18   1.08
```

A negative Tajima's D signifies an excess of low frequency polymorphisms, indicating population size expansion. While a positive Tajima's D indicates low levels of both low and high frequency polymorphisms, thus a sign of a population bottleneck. In constant size populations, Tajima's D is expected to be zero.

The data set `human` contains four objects, and `stat.voight` is just one of them. The manual page of all objects can be accessed by referring to one of its objects that we can achieved by typing for example, `?stat.voight`. Other objects of `human` contain the simulated summary statistics (`stat.3pops.sim`) and the model indices (`models`). Using these three objects, we can estimate the posterior probabilities of different demographic scenarios (see below) in the three human populations: Hausa, Italian, and Chinese. The fourth object, `par.italy.sim` may be used to estimate the ancestral population size of the European population (represented by an Italian sample).

The demographic models

Since we do not know which demographic model would be the most appropriate to estimate the ancestral human population size, first, we propose three different models to see which model is the most supported by the data. The three models of demographic history are: constant population size, exponential growth after a period of constant population size, and population bottleneck, where, after the bottleneck, the population recoveres to its original size. All three models can be defined by a number of demographic parameters, such as population sizes, rates and timings of changes in population size, which we do not detail here. The data has been generated for the users, however, can easly be re-generated and the demographic parameters may be altered. We simulated $50,000$ data sets under each of the three demographic models using the software `ms` (Hudson, 2002), and calculated the three summary statistics in each model. The exact R code to simulate the `human` data can be found in `/inst/doc/runms.R`, thus can be viewed, modified and re-run by interested users, given that `ms` and its sister package `sample_stats` are installed and configured.

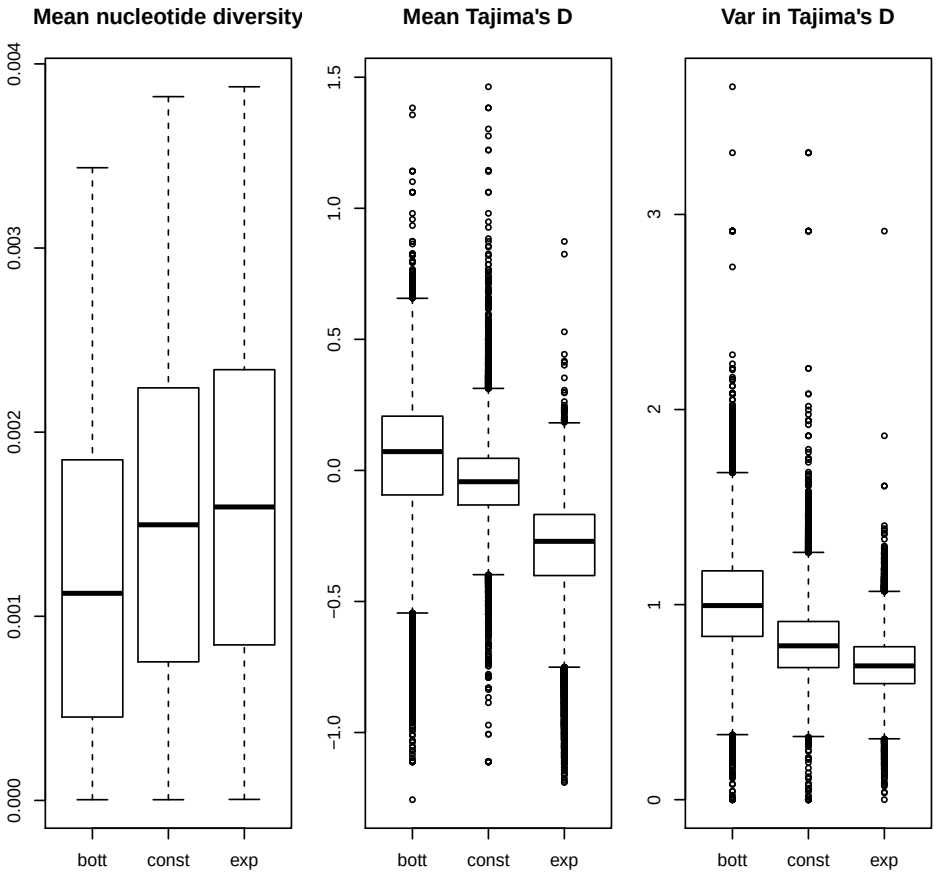

Figure 1: Summary statistics under the three demographic models: population bottleneck, constant population size, exponential growth

To illustrate the informativeness of our summary statistics we can plot them as a function of the model (Figure 1).

```
> par(mfcol = c(1, 3), mar = c(5, 3, 4, 0.5))
> boxplot(stat.3pops.sim[, "pi"] ~ models, main = "Mean nucleotide diversity")
> boxplot(stat.3pops.sim[, "TajD.m"] ~ models, main = "Mean Tajima's D")
> boxplot(stat.3pops.sim[, "TajD.v"] ~ models, main = "Var in Tajima's D")
```

Model selection

The main model selection function is called `postpr`. However, before applying this function on the real data, we perform a cross-validation for model selection (`cv4postpr`) to evaluate if ABC can, at all, distinguish between the three models. The cross-validation might take a long time to run. At this

point, we might just want just a quick result to illustrate the use of the function, so we run only 50 cross-validation simulations, and summarize and the results using the following commands:

```
> cv.modsel <- cv4postpr(models, stat.3pops.sim, nval = 50, tol = 0.01,
+     method = "mnlogistic")
> s <- summary(cv.modsel)

Confusion matrix based on 50 samples for each model.

$tol0.01
      bott const exp
bott    37    12   1
const    5    36   9
exp      2     7  41


Mean model posterior probabilities (mnlogistic)

$tol0.01
        bott  const    exp
bott  0.7045 0.2178 0.0777
const 0.2154 0.5488 0.2358
exp   0.0603 0.2045 0.7352
```

The resulting confusion matrix may also be plotted using the following command:

```
> plot(cv.modsel, names.arg = c("Bottleneck", "Constant", "Exponential"))
```

Both the confusion matrix and the barplot (Figure 2) illustrate that ABC is able to distinguish between these three models. Notably, the exponential expansion model can be classified correctly the most frequently: 41 times out of 50. However, the total misclassification rate of 0.24 prompts us to be cautious about the results obtained with model selection and suggests that additional data or summary statistics are required to better discriminate between the models. Remember that simulations are randomly selected, so every user will get slightly different figures for the misclassification rates, especially if `nval` is small.

Now, we may calculate the posterior probabilities of each demographic scenario using the rejection (`"rejection"`) and the multinomial logistic regression method (`"mnlogistic"`) of the function `postpr` with a tolerance rate of 0.05%. The function `summary` prints out posterior model probabilities and ratios of model probabilities (the Bayes factors) in a user-friendly way:

```
> modsel.ha <- postpr(stat.voight["hausa", ], models, stat.3pops.sim,
+     tol = 0.05, method = "mnlogistic")
> modsel.it <- postpr(stat.voight["italian", ], models, stat.3pops.sim,
+     tol = 0.05, method = "mnlogistic")
> modsel.ch <- postpr(stat.voight["chinese", ], models, stat.3pops.sim,
+     tol = 0.05, method = "mnlogistic")
> summary(modsel.ha)

Call:
postpr(target = stat.voight["hausa", ], index = models, sumstat = stat.3pops.sim,
```

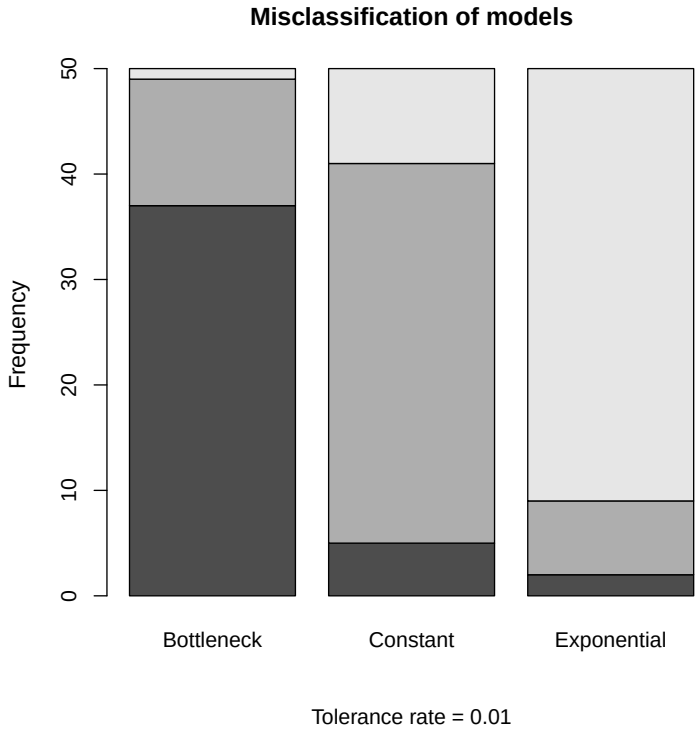

Figure 2: Misclassification proportions for the tree models. If all tree models were classified correctly 100% of the times, all three bars would have a single colour.

```
    tol = 0.05, method = "mnlogistic")
Data:
 postpr.out$values (7500 posterior samples)
Models a priori:
 bott, const, exp
Models a posteriori:
 bott, const, exp

Proportion of accepted simulations (rejection):
  bott  const    exp
0.0199 0.3132 0.6669

Bayes factors:
         bott   const     exp
bott   1.0000  0.0634  0.0298
const 15.7651  1.0000  0.4696
exp   33.5705  2.1294  1.0000


Posterior model probabilities (mnlogistic):
  bott  const    exp
0.0164 0.3591 0.6245

Bayes factors:
         bott   const     exp
bott   1.0000  0.0456  0.0262
const 21.9360  1.0000  0.5751
exp   38.1446  1.7389  1.0000

> summary(modsel.it)

Call:
postpr(target = stat.voight["italian", ], index = models, sumstat = stat.3pops.sim,
    tol = 0.05, method = "mnlogistic")
Data:
 postpr.out$values (7500 posterior samples)
Models a priori:
 bott, const, exp
Models a posteriori:
 bott, const, exp

Proportion of accepted simulations (rejection):
  bott  const    exp
0.8487 0.1509 0.0004

Bayes factors:
           bott     const       exp
bott     1.0000    5.6228 2121.6667
const    0.1778    1.0000  377.3333
exp      0.0005    0.0027    1.0000
```

```
Posterior model probabilities (mnlogistic):
  bott  const    exp
0.9369 0.0628 0.0004

Bayes factors:
           bott     const       exp
bott     1.0000   14.9246 2508.9790
const    0.0670    1.0000  168.1105
exp      0.0004    0.0059    1.0000

> summary(modsel.ch)

Call:
postpr(target = stat.voight["chinese", ], index = models, sumstat = stat.3pops.sim,
    tol = 0.05, method = "mnlogistic")
Data:
 postpr.out$values (7500 posterior samples)
Models a priori:
 bott, const, exp
Models a posteriori:
 bott, const, exp

Proportion of accepted simulations (rejection):
  bott  const    exp
0.6837 0.3159 0.0004

Bayes factors:
           bott     const       exp
bott     1.0000    2.1646 1709.3333
const    0.4620    1.0000  789.6667
exp      0.0006    0.0013    1.0000


Posterior model probabilities (mnlogistic):
  bott  const    exp
0.7610 0.2389 0.0001

Bayes factors:
            bott      const        exp
bott      1.0000     3.1853 10840.7807
const     0.3139     1.0000  3403.3749
exp       0.0001     0.0003     1.0000
```

These results shwo that Hausa data supports best the model of exponential growth model, while the Italian and Chinoise data supports best the population bottleneck model.

Now we can move to our original objective, which is to estimate the ancestral population size. To do so, in the following sections we will focus on only one of the population data. We choose the data set from Italy, for which we will estimate the ancestral population size under the (most supported) bottleneck model.

Posterior predictive checks

To further confirm that the bottleneck model provided the best fit to the Italian data, we considered posterior predictive checks Gelman et al. (2003). Note that there is no specific function in `abc` for posterior predictive checks, nevertheless the task can be easily carried out. First, we estimate the posterior distributions of the parameters of the three models using the function `abc` (see details in the following section). Then, we sample a set of 1,000 multivariate parameters from their posterior distribution. Last, we obtain a sample from the distribution of the three summary statistics a posteriori by simulating data sets with the 1,000 sampled multivariate parameters using `ms`. The exact code to run `ms` again can be found in `\inst\doc\runms4ppc.R`. By running this code one can re-generate the package's data set `ppc`.

The results of the posterior predictive checks could best be dispayed as histograms:

```
> data(ppc)
> mylabels <- c("Mean nucleotide diversity", "Mean Tajima's D",
+     "Var Tajima's D")
> par(mfrow = c(1, 3), mar = c(5, 2, 4, 0))
> for (i in c(1:3)) {
+     hist(post.bott[, i], breaks = 40, xlab = mylabels[i], main = "")
+     abline(v = stat.voight["italian", i], col = 2)
+ }
```

The Figure 3 illustrates the posterior predicitive checks under the bottleneck model. We can see that the bottleneck model is able to reproduce the observed values of the summary statistics in the Italian data. Note that the constant population size model is also able to reproduce the observed summary statistics of the Italian data. However, the model of exponential growth is unable to account for the observed level of heterozygosity when accounting for the remaining two summary statistics in the Italian sample (results not shown, but maybe generated by the user).

Finally, note that such posterior predictive checks use the summary statistics twice; once for sampling from the posterior and once for comparing the marginal posterior predictive distributions to the observed values of the summary statistics. To avoide this circularity, we might consider using different summary statistics for posterior predictive checks than for parameter estimation. Nevertheless, this could be difficult in many applications, since we already used our "best" summary statistics for inference.

Cross-validation

Now, we are almost ready to infer the ancestral population size under the bottleneck model for the Italian data. So, we select the simulated parameter values that correspond to the bottleneck model:

```
> stat.italy.sim <- subset(stat.3pops.sim, subset = models == "bott")
> head(stat.italy.sim)
```

```
                 pi      TajD.m    TajD.v
110002 0.0010531112 0.057649360 1.3024486
210000 0.0012026666 0.342608757 1.1042096
310000 0.0008920002 0.247601809 0.9147642
410002 0.0015328889 0.470767633 1.5339313
51002  0.0008988888 0.008920568 1.0751214
61002  0.0019851108 0.208063980 0.8684532
```

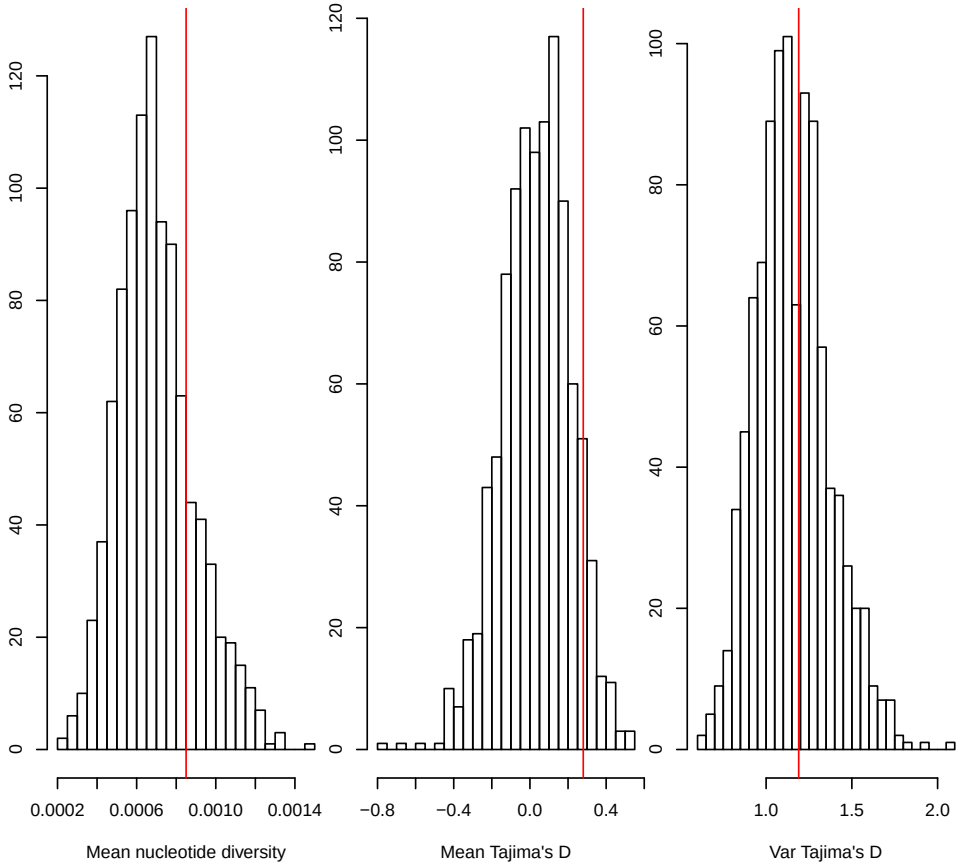

Figure 3: Posterior predictive checks for the Italian data under the bottleneck model

The bottleneck model can be described with four parameters, which can be found in the four columns of `par.italy.sim`: the ancestral population size N_a, the ratio of the population sizes before and during the bottleneck (`a`), the duration of the bottleneck (`duration`), and the time since the beginning of the bottleneck (`start`).

```
> head(par.italy.sim)

         Ne        a duration    start
1 14621.380 28.58894 5428.309 40421.04
2 13098.281 18.65774 5081.701 43255.28
3  7936.504 12.00553 3369.698 50781.08
4 17823.659 42.57813 7857.355 46397.16
5 12294.555 23.60331 6633.745 58208.44
6 25626.369 40.09893 7067.745 50385.51
```

Before moving to the inference step, we first assess if ABC is able to estimate the parameter N_a at all. We use the function `cv4abc` to determine the accuracy of ABC and the sensitivity of estimates to the tolerance rate. The following code evaluates the accuracy of estimates of N_a under three tolerance rates using the rejection and the local linear regression method. The `"neuralnet"` method is not recommended for cross-validation, as it takes considerably longer to run than the other methods. Since the cross-validation may take a long time to run, we might prefer to run first a code with a smaller value of `nval` to compare different tolerance rates, and also different estimation methods.

```
> cv.res.rej <- cv4abc(data.frame(Na = par.italy.sim[, "Ne"]),
+     stat.italy.sim, nval = 200, tols = c(0.005, 0.01, 0.05),
+     method = "rejection")
> cv.res.reg <- cv4abc(data.frame(Na = par.italy.sim[, "Ne"]),
+     stat.italy.sim, nval = 200, tols = c(0.005, 0.01, 0.05),
+     method = "loclinear")
> summary(cv.res.rej)

Prediction error based on a cross-validation sample of 200

              Na
0.005 0.07885375
0.01  0.09177791
0.05  0.12872101

> summary(cv.res.reg)

Prediction error based on a cross-validation sample of 200

              Na
0.005 0.05068098
0.01  0.05017856
0.05  0.05059759
```

We can also plot the results using the following code:

```
> par(mfrow = c(1, 2), mar = c(5, 3, 4, 0.5), cex = 0.8)
> plot(cv.res.rej, caption = "Rejection")
> plot(cv.res.reg, caption = "Local linear regression")
```

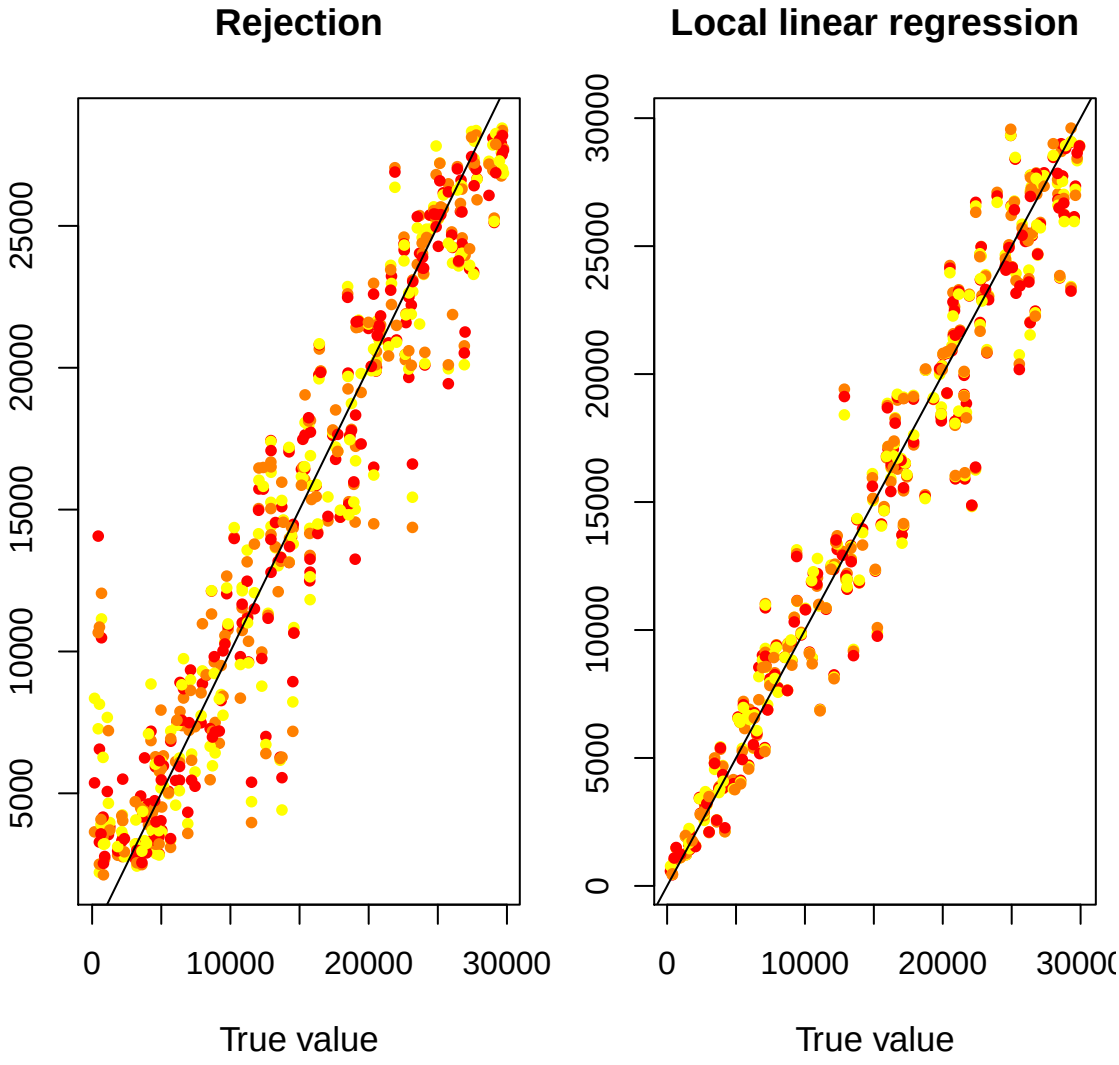

Figure 4: Cross-validation for parameter inference. The colors correspond to different levels of the tolerance rate in an increasing order from red to yellow.

The function `summary` shows the prediction error under the three different tolerance rates. `plot` compares the two methods (`"rejection"` and `"loclinear"`) under three different tolerance rates. Users can choose how they want to summarize the posterior distribution by setting the argument `statistic` in of the function `cv4abc`. By default, the posterior distribution of summarized by its median. Thus, the plots shows the posterior distribution medians of N_a for each cross-validation sample. Points of the cross-validation plot are scattered around the identity line indicating that N_a can be well estimated using the three summary statistics (Figure 4). Further, estimates were not only accurate for N_a, but also insensitive to the tolerance rate (Figure 4). Accordingly, the prediction error is relatively low and independent of the tolerance rate.

Parameter inference

Now, we finaly estimate the posterior distribution of N_a using the main function of the package: `abc`. The following code shows how to use `abc` with the method `"neuralnet"`. We chose to use a `"log"` transformation of the parameter. The correction for heteroscedasticity of performed by default.

```
> res <- abc(target = stat.voight["italian", ], param = data.frame(Na = par.italy.sim[,
+     "Ne"]), sumstat = stat.italy.sim, tol = 0.05, transf = c("log"),
+     method = "neuralnet")

12345678910
12345678910

> res

Call:
abc(target = stat.voight["italian", ], param = data.frame(Na = par.italy.sim[,
    "Ne"]), sumstat = stat.italy.sim, tol = 0.05, method = "neuralnet",
    transf = c("log"))
Method:
Non-linear regression via neural networks
with correction for heteroscedasticity

Parameters:
Na

Statistics:
pi, TajD.m, TajD.v

Total number of simulations 50000

Number of accepted simulations:  2500
```

The function `abc` returns an object of class `"abc"`. The function `print` returns a simple summary of the object (see above). Using the function `summary` we can calculate summaries of the posterior distributions, such as the mode, mean, median, and credible intervals, taking into account the posterior weights, when appropriate. We obtain the following summary of the posterior distribution using the function `summary`:

```
> summary(res)
```

```
Call:
abc(target = stat.voight["italian", ], param = data.frame(Na = par.italy.sim[,
    "Ne"]), sumstat = stat.italy.sim, tol = 0.05, method = "neuralnet",
    transf = c("log"))
Data:
 abc.out$adj.values (2500 posterior samples)
Weights:
 abc.out$weights

                              Na
Min.:                   7480.835
Weighted 2.5 % Perc.:   8906.354
Weighted Median:       11463.221
Weighted Mean:         11729.197
Weighted Mode:         11076.389
Weighted 97.5 % Perc.: 15796.591
Max.:                  19798.780
```

Two functions are available to visualize the results, `hist` and `plot`. `hist` simply displays a histogram of the weighted posterior sample (Figure 5).

```
> hist(res)
```

The function `plot` can be used as a diagnostic tool when one of the regression methods is applied. We can run the following code to generate the diagnostic plots of the estimation of the posterior distribution of N_a:

```
> par(cex = 0.8)
> plot(res, param = par.italy.sim[, "Ne"])
```

Figure 6 shows that `plot` returns various plots that allow the evaluation of the quality of estimation when one of the regression methods is used. The following four plots are generated: a density plot of the prior distribution, a density plot of the posterior distribution estimated with and without regression-based correction, a scatter plot of the Euclidean distances as a function of the parameter values, and a normal Q-Q plot of the residuals from the regression. When the heteroscedastic regression model is used, a normal Q-Q plot of the standardized residuals is displayed.

Figure 6 shows that the posterior distribution is very different from the prior distribution, confirming that the three summary statistics convey information about the ancestral population size (Figure 6, lower left panel). The upper right panel of Figure 6 shows the distance between the simulated and observed summary statistics as a function of the prior values of N_a. Again, this plot confirms that the summary statistics convey information about N_a, because the distances corresponding to the accepted values are clustered and not spread around the prior range of N_a. The lower right panel displays a standard Q-Q plot of the residuals of the regression, $\hat{\epsilon}_i$. This plot serves as a regression diagnostic of the linear or non-linear regression when the method is `"loclinear"` or `"neuralnet"`.

Conclusions

A few of the capabilities of the `abc` package have been described. Inevitably, new applications will demand new features and might reveal unforeseen bugs. Please send comments or suggestions and report bugs to `kati.csillery@gmail.com` and/or `michael.blum@imag.fr`.

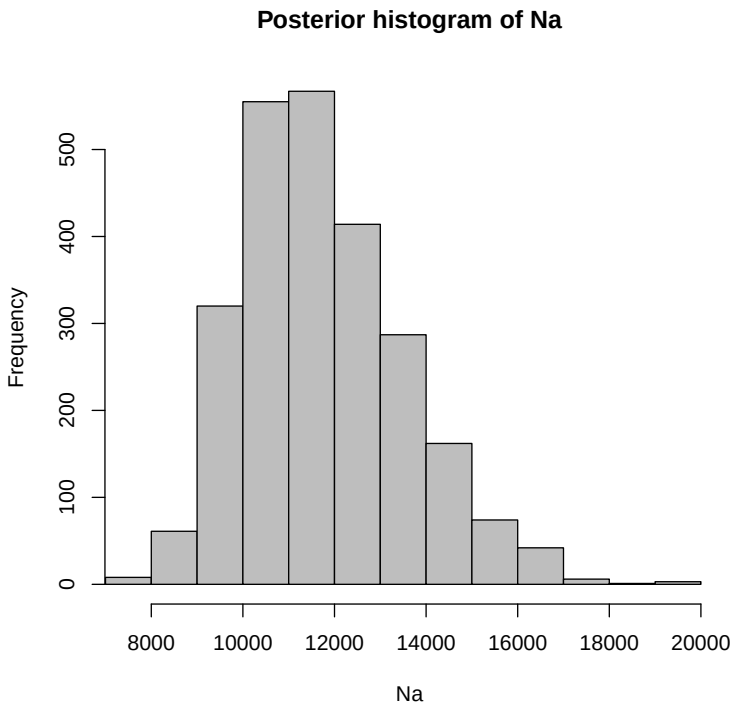

Figure 5: Histogram of the weighted posterior sample of N_a

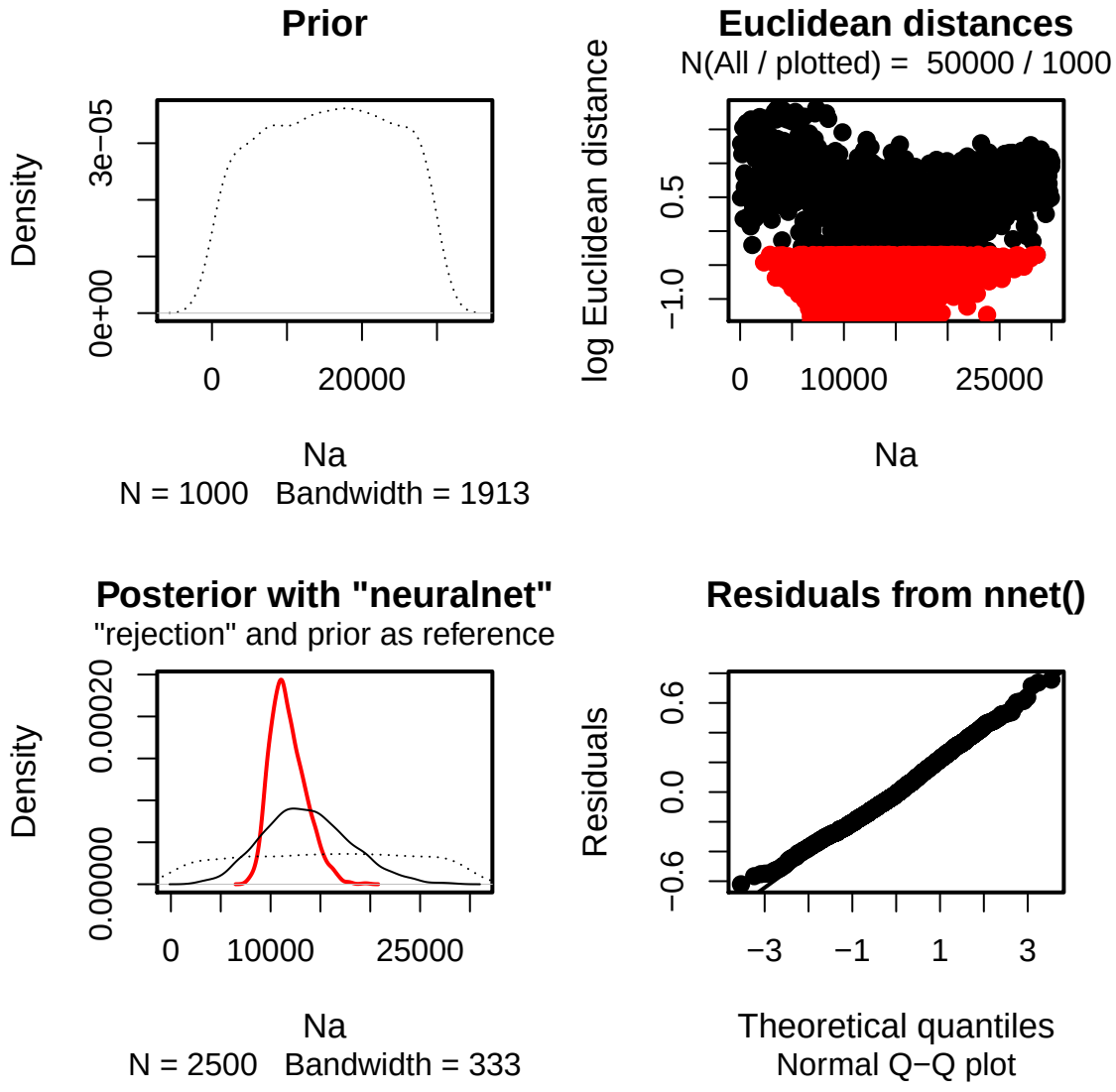

Figure 6: ABC regression diagnostics for the estimation of the posterior distribution of N_a

References

M A Beaumont. Joint determination of topology, divergence time, and immigration in population trees. In Renfrew C Matsumura S, Forster P, editor, *Simulation, Genetics and Human Prehistory*, McDonald Institute Monographs, pages 134–1541. McDonald Institute Monographs, UK, 2008.

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

Mark A. Beaumont, Jean-Marie Cornuet, Jean-Michel Marin, and Christian P. Robert. Adaptive approximate bayesian computation. *Biometrika*, 96(4):983–990, 2009.

M G B Blum and O François. Non-linear regression models for Approximate Bayesian Computation. *Statistics and Computing*, 20:63–73, 2010.

K Csilléry, M G B Blum, O E Gaggiotti, and O François. Approximate Bayesian Computation in practice. *Trends Ecol Evol*, 25:410–418, 2010. doi: 10.1016/j.tree.2010.04.001.

N.J.R. Fagundes, N. Ray, M. 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*, 104(45):17614, 2007.

Olivier François and Guillaume Laval. Deviance information criteria for model selection in approximate bayesian computation. *Statistical Applications in Genetics and Molecular Biology*, 10(1):33, 2011.

Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis, Second Edition (Texts in Statistical Science)*. Chapman & Hall/CRC, Boca Raton, 2 edition, 2003.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, 2009.

Richard R. Hudson. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics*, 18(2):337–338, 2002. doi: 10.1093/bioinformatics/18.2.337.

Guillaume Laval and Laurent Excoffier. SIMCOAL 2.0: a program to simulate genomic diversity over large recombining regions in a subdivided population with a complex history. *Bioinformatics*, 20 (15):2485–2487, 2004.

Paul Marjoram, John Molitor, Vincent Plagnol, and Simon Tavaré. Markov chain Monte Carlo without likelihoods. *Proc Natl Acad Sci USA*, 100(26):15324–15328, 2003.

P. Pavlidis, S. Laurent, and W. Stephan. msabc: a modification of hudson's ms to facilitate multilocus abc analysis. *Molecular Ecology Resources*, 10(4):723–727, 2010. URL http://dx.doi.org/10.1111/j.1755-0998.2010.02832.x.

J K Pritchard, M T Seielstad, A Perez-Lezaun, and M W Feldman. Population growth of human Y chromosomes: a study of y chromosome microsatellites. *Molecular Biology and Evolution*, 16: 1791–1798, 1999.

B. F. Voight, A. M. Adams, L. A. Frisse, Y. Qian, R. R. Hudson, and A. Di Rienzo. Interrogating multiple aspects of variation in a full resequencing data set to infer human population size changes. *Proc Natl Acad Sci USA*, 102:18508–18513, 2005.

Daniel Wegmann, Christoph Leuenberger, Samuel Neuenschwander, and Laurent Excoffier. Abctoolbox: a versatile toolkit for approximate bayesian computations. *BMC Bioinformatics*, 11(1):116, 2010. ISSN 1471-2105. doi: 10.1186/1471-2105-11-116. URL http://www.biomedcentral.com/1471-2105/11/116.