Bringing ABC inference to the machine learning realm : AbcRanger, an optimized random forests library for ABC

François-David Collin, Arnaud Estoup, Jean-Michel Marin, Louis Raynal

Bringing ABC inference to the machine learning realm : AbcRanger, an optimized random forests library for ABC

François-David Collin[1], Arnaud Estoup[2], Jean-Michel Marin[1] and Louis Raynal[1]
[1] Université de Montpellier, CNRS, IMAG UMR 5149, Montpellier, France
[2] CBGP, INRA, CIRAD, IRD, Montpellier SupAgro, Univ. Montpellier, Montpellier, France

Corresponding author: Francois-David.Collin@umontpellier.fr

Abstract *The AbcRanger library provides methodologies for model choice and parameter estimation based on fast and scalable* Random Forests*, tuned to handle large and/or high dimensional datasets. The library, initially intended for the population genetics ABC framework DIYABC, has been generalized to any ABC reference table generator.*

At first, computational issues were encountered with the reference ABC-Random Forest. Those issues have been diagnosed by us as friction between "strict" Machine Learning setup and ABC context, and this incited us to modify the C++ implementation of state-of-the-art random forests, ranger, to tailor it for ABC needs: potentially "deep" decision trees are not stored in memory anymore, but are processed by batches in parallel.

We focused on memory and thread scalability, ease of use (minimal hyperparameter set). R and python interfaces are provided. A toy example and a population genetics example are presented.

Keywords Approximate Bayesian Computation, Random Forests, Model Choice, Parameter Estimation, C++, Python, R

1 Introduction : challenges for ABC from Population Genetics

In the context of recent advances in population genetics, the *number of simulated data* in a ABC context could reach over the hundred of thousands (10^5) mark. Similarly, with the advent of multi-population summary statistics in this domain (see [1]) the number of summary statistics computed by ABC (as covariables) could range from several hundreds to tens of thousands (scenario with several populations and combinatorial "explosion" of multi-population statistics). Moreover, not all summary statistics are relevant, and traditional variable selection methods still have to be tuned for each case in an *ad hoc* manner. From both row and column inflations point of view, classical methods for ABC (k-nn and local methods) don't cope very well with this situation.

[2] and [3] proposed a novel approach, coined as *ABC-random forest* or *ABC-RF*, which relies on *Random Forests* to provide tractable and efficient methodologies, for both model choice and parameter estimation.

2 First building block : ABC simulations to generate the Random Forest training database

In a Bayesian context, when the likelihood function is too complex or untractable, several *likelihood-free* methods are available to approximate it, including Approximate Bayesian Computation (ABC) [4]. Given observed data, the basic idea of ABC is to approximate the likelihood of a parameterized model with selected simulations, by comparing the observed data and simulated ones via computed *summary statistics*. The table of summary statistics for simulated data is called *the reference table* (see 1). It corresponds to the so called "training dataset" in Machine Learning terminology.

2.1 ABC posterior methodologies

2.1.1 Model Choice Given an observed data, and several (parameterized) models, the purpose is to estimate the best model to fit our data. A reference table combining summary statistics of simulated samples (particles) is generated from each model (models are sampled according to a prior distribution, e.g. by penalizing the model complexity). A *Model Choice* methodology is an inference method which takes this reference table, the observed data and *infers* the best fitted model for this

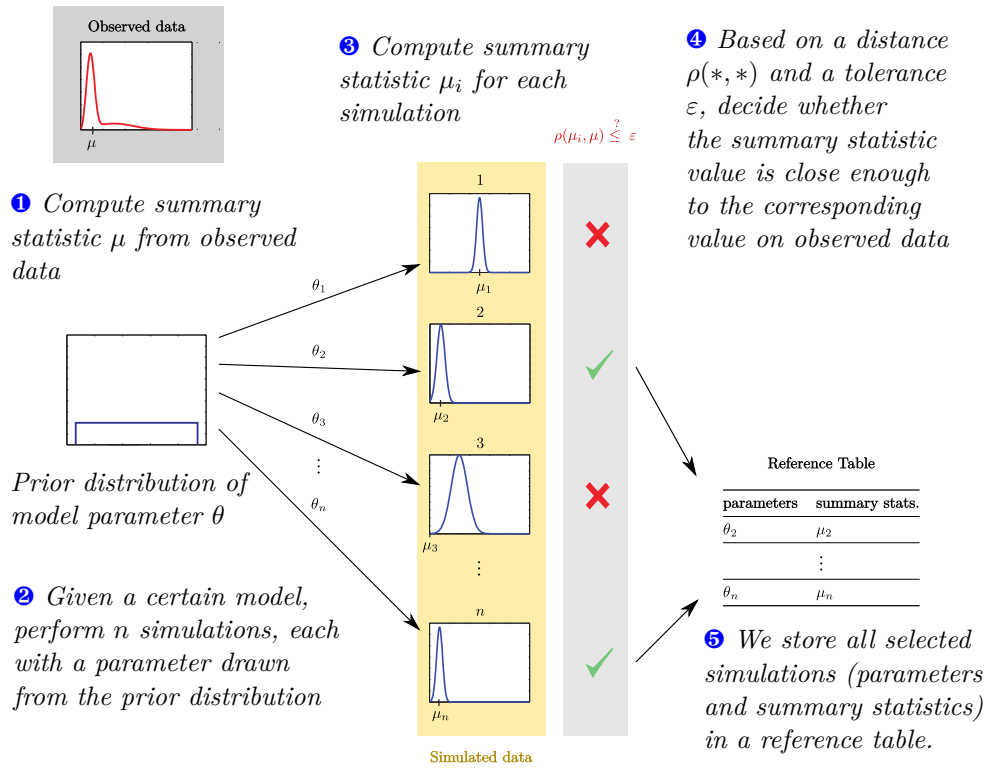

Fig. 1. ABC simulations to generate the Random Forest training database

data, along with an estimated posterior probability (the probability of the model knowing the observed data), which assesses the fitness of the predicted model.

2.1.2 Parameter Estimation Given an observed data and one parameterized model, the purpose is to infer one or several parameters for this model given the observed data. An ABC reference table is generated from the model. The *Parameter estimation* methodology is an inference method which takes this reference table, the observed data and *infers* one or several parameters, along with the usual Bayesian decorum : posterior distribution, quantiles and so on.

2.1.3 General workflow A sensible workflow is to first choose a model and then infer its parameters (see 2).

3 Second building block : Random Forests

Enter the *Supervised Machine Learning* (SML) realm [5]: at the beginning lies a list of pairs of input data/output data $\{x_i, y_i\}$ from (X, Y) domain, called a *training dataset* (the output is also called the *target*). The objective is to *learn* the best function $f_\theta(x)$ parameterized by $\theta \in \Theta$ [5] so that a *scalar loss function* $L : Y \times Y \mapsto \mathbb{R}$ is minimized on the Θ domain :

$$f_\theta = \underset{\theta}{\operatorname{argmin}}\ L(f(x_i), y_i)$$

A specific class of learning function f_θ are Random Forests (RF), which we are going to expose. RF are based on CART, *Classification and Regression Trees*, an algorithm developed by [6].

3.1 CART

A CART is a *supervised machine learning algorithm* which essentially performs a partitioning of the predictor space into disjoint subspaces recursively. A prediction value is obtained and assigned

5. the θ parameter for the learning (fitting) function should not be confused with a *prior* θ parameter of a model in bayesian context.

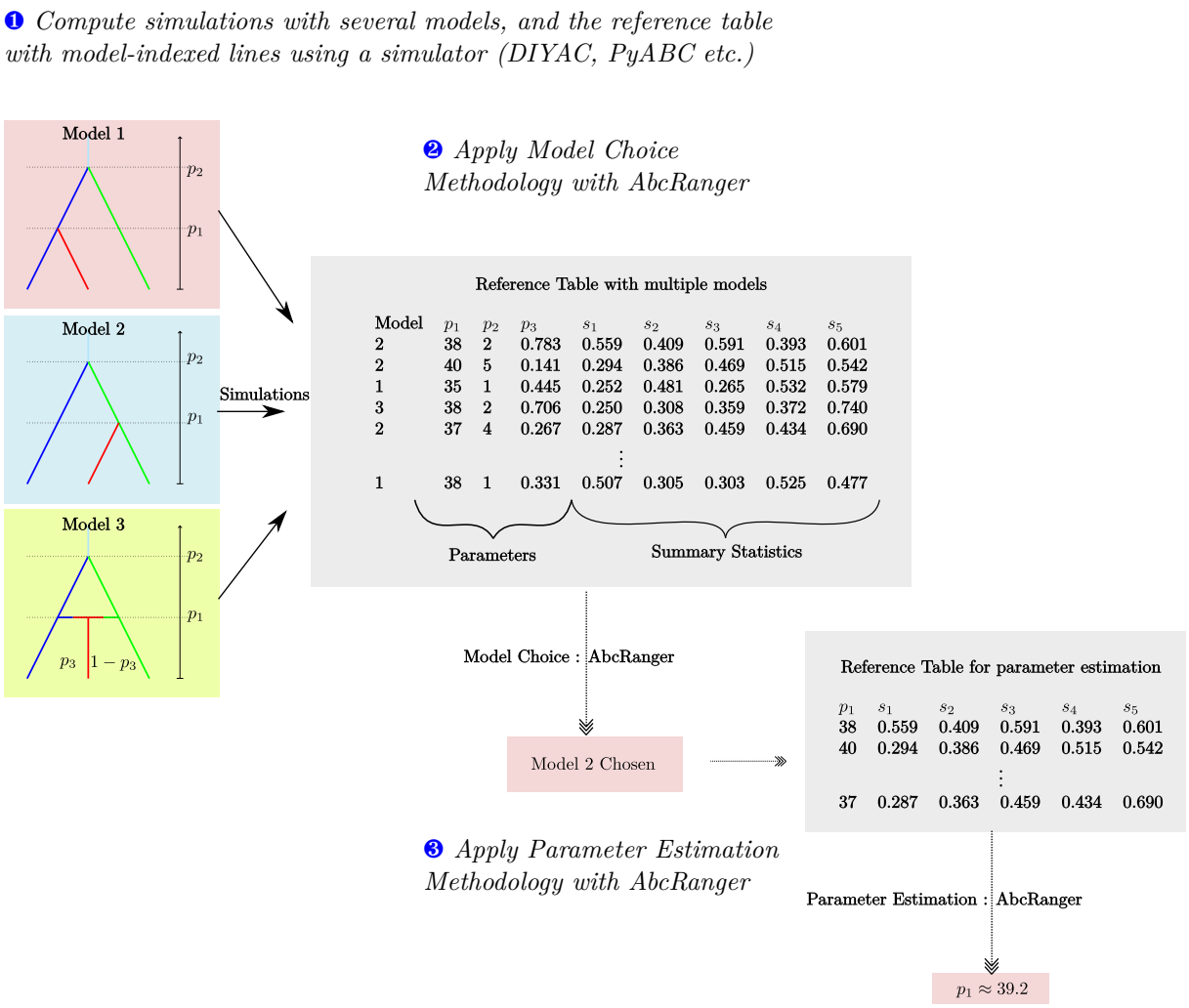

Fig. 2. Workflow with AbcRanger

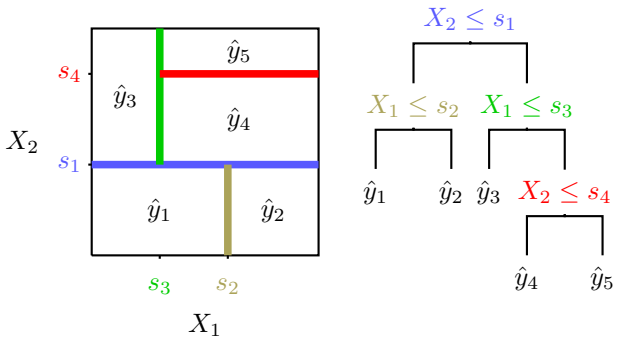

Fig. 3. An example of CART and the associated partition of the two dimensional predictor space. Each splitting condition takes the form $X_j \leq s$ and the prediction at a leaf is denoted $\hat{y}_\ell$.

to each of those subspaces (or *Leaves*), computed from the target (output) values of the samples contained in the corresponding leaves. Once the partitioning is done, the result is a binary tree which could predict outcomes from an input data, either classes or continuous values, by *routing* the data to a *leaf*, whose assigned value will be used then as prediction (see 3).

3.2 Random Forests

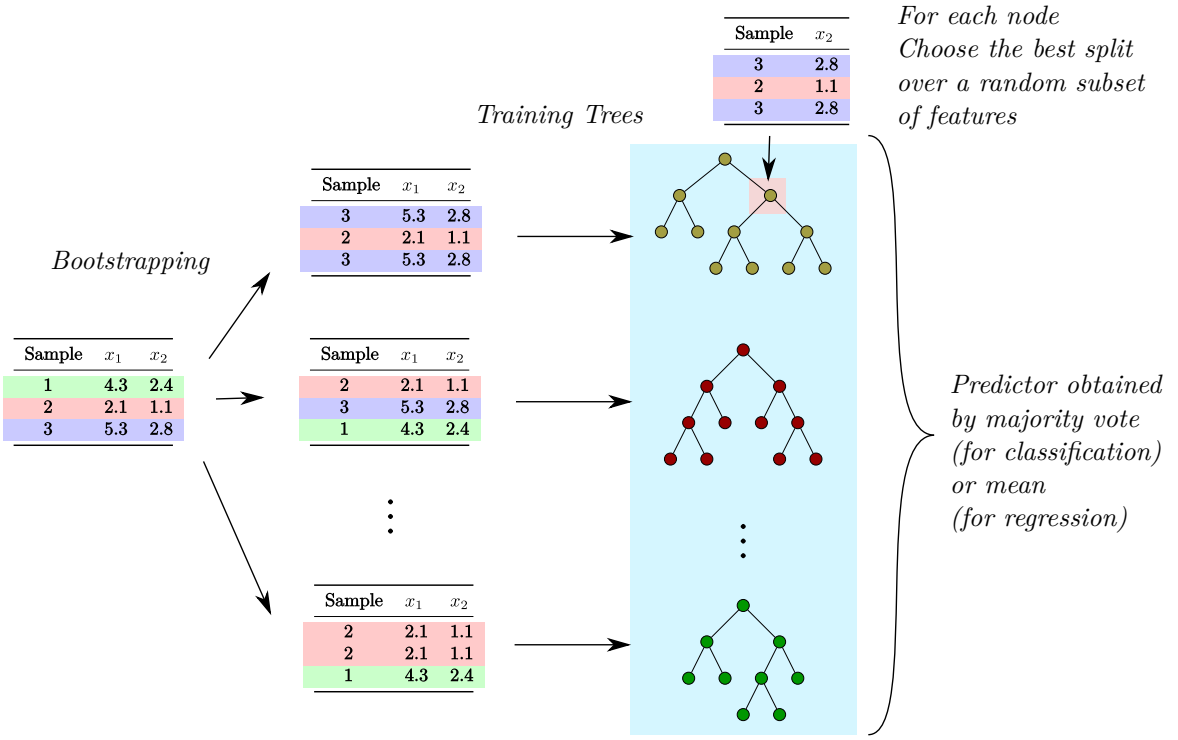

Fig. 4. Random Forest

Random Forests [7] are a three pronged extension of CART (see 4). First, it is an **Ensemble method** which trains a *set* of CART (not just one) and predict the outcome with the majority vote (resp. mean) of this set of trained trees for classification (resp. regression) target (or output data). Second, **bootstrapping** is applied before each tree training, i.e. training data is randomly sampled (with replacement). And last but not least, in a growing tree, at each node, the best split is computed on a **random subset of the features**. Those three extensions have multiple benefits; the main ones are lower variance compared to a single CART tree, due to the ensemble method, and *unbiasedness*, because of the de-correlation of the trees induced by both bootstrapping and random sampling of features. Other advantages are: robustness to noise, variable importance for (almost) free, integrated cross-validation procedure (out-of-bag samples, no need to get a validation dataset), easy parallelization, very good scaling properties (both in rows and columns axes), and availability for both classification and regression targets.

3.3 ABC Random Forest

A reference implementation of the *ABC-Random Forest* setup is given by *abcrf* [8]. Here, we provide a brief description of ABC Random Forest methodologies for model choice and parameter estimation. The input in both case is the reference table of summary statistics obtained by ABC, the predictor space is the models (discrete) in case of model choice or a specific model parameter for

3.3.1 Model Choice Model Choice methodology in ABC-RF is two staged. *First a classification random forest is trained* with the models (classes) as target. The trained random forest model is evaluated on the observed data, getting votes and the best model to fit. *Second*, using the obtained random forest from the first stage, each sample from the training dataset is labeled classified/misclassified with the *out-of-bag prediction* and finally as numerical 0 or 1 for a new target. Then, a *new regression*

random forest is trained on the training dataset, but this time with this new target (as continuous, non-categorical one for regression). And finally a prediction on the observed data is evaluated with the obtained random forest, and this predicted value (between 0 and 1) is a viable estimator for the *posterior probability of the chosen model*.

3.3.2 Parameter Estimation In ABC Random Forest setup, parameter estimation is limited to one parameter at a time. Choosing a parameter θ to estimate, a regression RF is trained on a reference table generated only with the corresponding model and with the θ parameter values as target, forming the training dataset. Once trained, the regression RF is evaluated on the observed data and several outcomes are obtained, like an estimation of θ, variance, and quantiles with the help of **quantile regression forests** [9]. It is worth noting that Quantile Forests are not new forests per se but an – integrated – method to compute weights distribution of the samples, knowing an observed (or out-of-bag) data. This distribution is then used to compute quantiles, for example. Finally a set of both prior and posterior estimators is inferred from the RF predictions, for example a prior (resp. posterior) *pdf*, obtainable via standard kernel density estimation (resp. standard weighted density estimation).

3.4 Linear augmentations

As stated in [2] (resp. [3]), for Model choice (resp. parameter estimation), there is the option – enabled by default – to add linear combinations covariables to the existing summary statistics in the reference table via *Linear Discriminant Analysis* (resp. *Partial Least Squares*) [5]. By refining the partitioning of the trees [6], this sensibly improves the prediction accuracy of Random Forests outcomes.

3.5 Computational limitations with ABC Random Forests reference implementation

Faced with training dataset including 100 000 lines and more than 10 000 summary statistics, *abcrf* has been found growing trees over one gigabyte of memory size each. So, as typical random forests are made of 500 or 1000 trees for prediction performance, even with state of the art RF packages like [10], memory constraints are preventing completion of the training.

At first, limiting tree depth has been investigated but quickly dismissed because accuracy took an unacceptable hit.

This issue has a longer reach than an simple implementation issue and exhibits a fundamental mismatch of objectives between "classical" supervised machine learning setup and ABC posterior methodologies. Indeed, within "pure" SML, a model (like a Random Forest) is first trained, and then used to make predictions on a potentially endless source of new data; the whole model is stored by training and loaded in memory each time for prediction purpose. However, within the ABC inference context, the SML model is only needed for specific predictions directly on one or several observed data sample(s) and out-of-bag samples. Moreover, the corresponding trained Random Forest is coupled to the generated reference table (aka the training dataset), and is by no mean meant to generalize to new data (other reference tables), let alone other model and relevant observed data: in fact storing the forest is useless [7]. Those remarks established the need of an adaptation of random forest algorithm for ABC.

4 New implementation of Random Forest and ABC Random Forest

Based on our own version of the core RF (written in C++) from the ranger package [10], our new implementation of Random Forest for ABC, *AbcRanger*, solves the memory constraint issue related to the deep trees. Leveraging the cumulative nature of the ensemble method, Random Forest computations are now done in a *joint grow/predict phase* for each tree, and then optimized in order to grow a limited batch of trees in memory. As illustrated by fig. 5), this means that the computation of the growing and prediction steps for each tree is executed in a sequential –- i.e. batch-wise – order: as now tree growing and predictions are computed in a single pass, predictions and posteriors are then

6. In our previous bi-dimensional example 3, it is similar to add a linear combination of X_1 and X_2 to our input space, and the resulting partitioning may no longer be horizontally or vertically aligned partitions in the (X_1, X_2) predictor space.

7. the R package randomForest has indeed an option to discard the forest as soon the test predictions are done

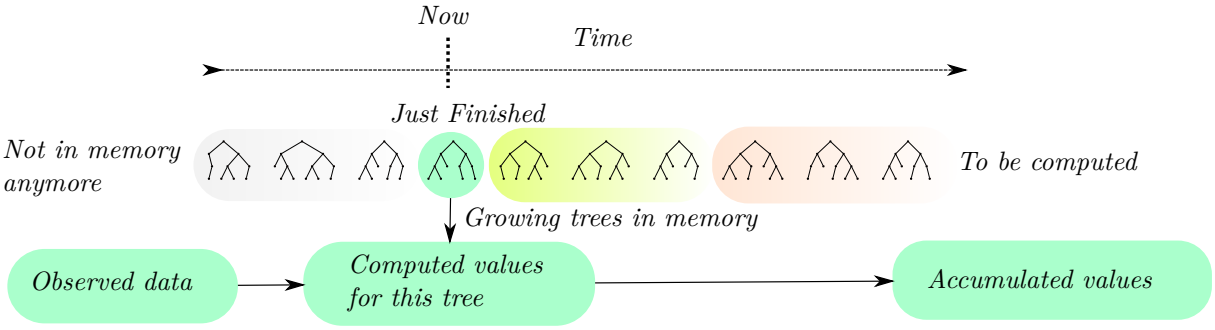

Fig. 5. Window of growing trees

stored/accumulated and each tree is finally discarded, freeing the system memory for next growing trees. The trees of the currently processed batch are still computed in parallel to leverage nowadays ubiquitous multicore architectures.

Although this doesn't precludes the in-memory storage of the entire training dataset at once, this way of processing avoids the in-memory storage of the whole forest at no performance cost. In a very constrained memory environment, one should just have to lower the number of computing threads to keep the memory of a training batch in check. A special care has also been taken to the Meinshausen's quantiles computations, completely parallelized and typically unnoticeable on multicore systems. Another advantage over *abcrf* package: methodologies are now pure C++. So, it is relatively easy to provide wrappers/interfaces to other languages than R, like Python, with the added guarantee that no copy of the reference table happens between the core C++ layer handling the methodology, and the interfaced language providing the reference table.

4.1 A toy example application with the ELFI python package

The *ELFI* python package [11] provides a popular and flexible ABC framework, meant to integrate complex ABC and inferences pipelines. Inspired by the $Ma(2)$ toy example used by original ABC authors in [4], we used a more general $Ma(q)$ example for model choice and parameter estimation, fixing $q = 10$ in the following.

$MA(q)$ is a time series called moving average model of order q defined by :

$$x_t = \mu + \epsilon_t - \sum_{i=1}^{q} \vartheta_i \epsilon_{t-i}$$

where μ is the mean of the series, the $\vartheta_1 ... \vartheta q$ the parameters of the model, and $\epsilon_t, \epsilon_{t-1}, ... \epsilon_{t-q}$ the white noise error terms. For identifiability purposes the parameters should verify the following condition, roots of

$$Q(u) = 1 - \sum_{i=1}^{q} \vartheta_i u^i$$

should be strictly outside the (complex) unit disc, and this is our main prior constraint. We choose another prior for the highest order parameter, θ_q, sampled from an uniform distribution (constraint domain).

From the generated examples of $Ma(10)$ on a 200-length signal, sampling the prior θ_q uniformly in the $[1, 2]$ interval, the usual row of (partial) autocorrelation features seems to be nonconclusive (see 6) to discriminate between, for example $Ma(8)$, $Ma(10)$ or $Ma(12)$.

4.1.1 Model choice: $Ma(10)$ vs "all" ($6 \leq q \leq 16$)

An ABC pipeline has been configured with *elfi*, choosing the default sampler, without rejection (option `quantile` fixed to 1). For 100 trials, starting with uniform distribution for θ_q, all other are constrained by the unity disc root exclusion

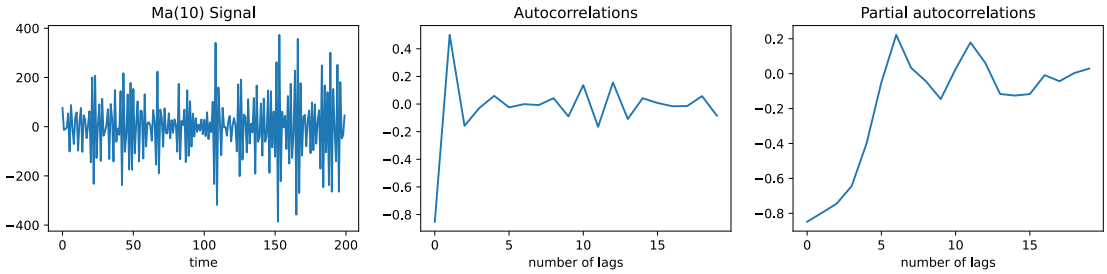

Fig. 6. Example of an $MA(10)$ model. From the left to the right, a $MA(10)$ signal, autocorrelations (correlation of the signal with its own delayed copy), and partial autocorrelations (the dependance of shorter delay removed).

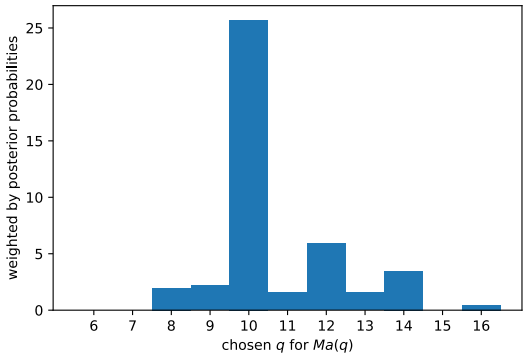

Fig. 7. Model Choice weighted histogram of inferred models: 100 $Ma(10)$ models are tried with ABC simulations followed by RF model choice inference (choosing from 10 different $MA(q)$ models where $6 \leq q \leq 16$, with signal length of 200 points and reftables of 2000 particles each).

described before, priors for $Ma(10)$ are sampled and an observation generated. Summary statistics are the autocorrelations, partial autocorrelations with 0.05, and 0.95 corresponding quantiles, for lags from 1 to 20. Models to choose are from $Ma(q)$ with $6 \leq q \leq 16$. On fig. 7, the performance of the ABC-RF setup is illustrated (python notebook).

Also, features coming from LDA linear augmentation are often discriminative, see fig. 8 for one inference example.

4.1.2 Parameter Estimation For parameter estimation one $Ma(10)$ is sampled and observed, and then all parameters are inferred individually with ABC-RF methodology (whith the help of *AbcRanger* python wrapper). Results are illustrated in fig. 9. All parameters of the model are nicely estimated, and the posterior/prior distributions clearly discriminated, except for the last θ_{10} parameter: this is, by essence, the most difficult parameter to infer, but it is expected as it is the highest order coefficient of a 10-polynomial.

4.2 A population genetics example

Reftables are generated by *DIYABC* simulator. DIYABC simulator with RF methodologies, as long than population genetics interpretations are presented in [12]. In this example, several evolutionary models with divergence and/or admixtures events are considered. Prior parameters are population sizes, times of events and the admixture rate (respective proportions of genes from admixed populations). Observed data is generated from the third model (see fig. 10).

The reftable contains a range of summary statistics (130, note that is a relatively low number, our experiments can achieve more than 20000), related to population genetics various measures. Abcranger selected the right model with a posterior probability around 70% (See fig. 11).

For parameter estimation of the admixture rate in model 3, we use another generated reftable, see fig. 12 and 13, for posterior inferences. The real value of the parameter is within its inferred

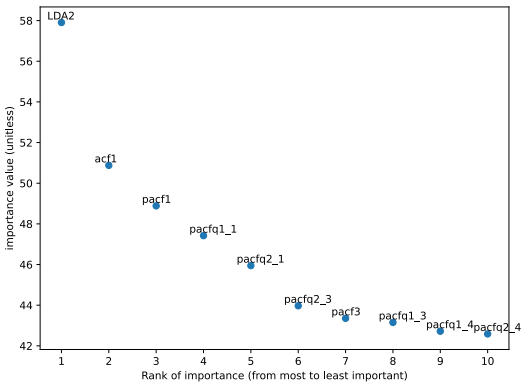

Fig. 8. 10 most ranked summary statistics, sorted by permutation importance. acf_i, *(resp.* $pacf_i$, $pacf1_i$, $pacf2_i$*)* are i-lagged autocorrelations (resp. partial autocorrelations, 0.05 and 0.95 corresponding quantiles).

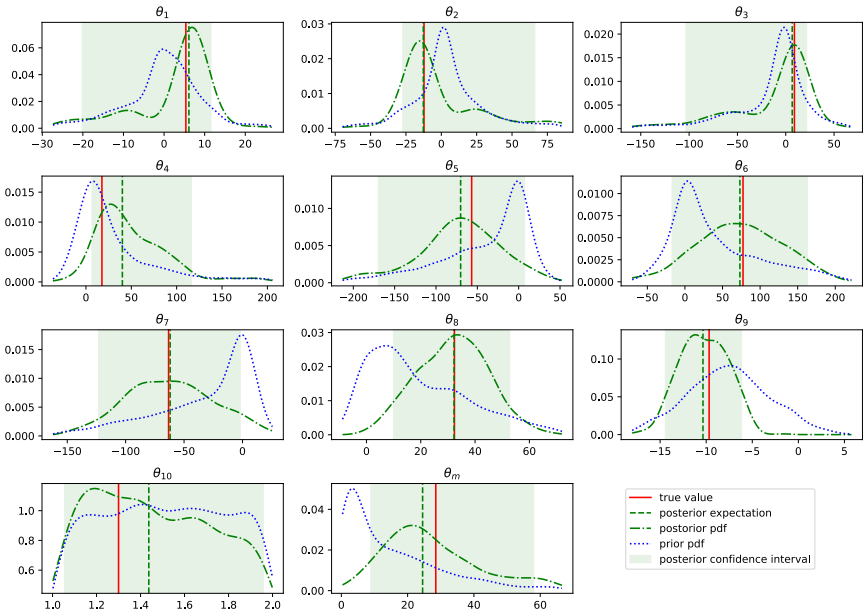

Fig. 9. Inferred posterior distributions of all prior parameters of the $MA(10)$ model. The θ_m is a dependant prior, computed mean of all θ_k priors ($1 \leq k \leq 10$).

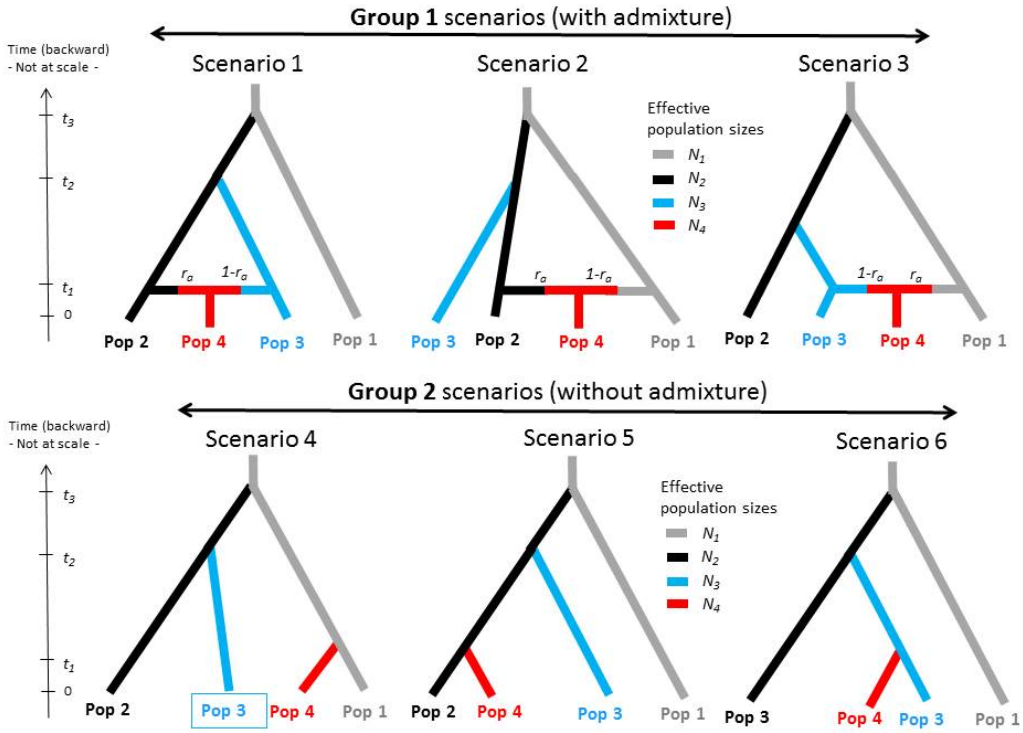

Fig. 10. Six evolutionary models considered for model choice, observed data comes from the third one. All scenarios get three times parameters (t_1, t_2 and t_3) and four population size parameters (N_1, N_2, N_3 and N_4). Scenerios 1, 2 and 3 have an additional ra admixture rate parameter for an admixture event. (illustration taken from [12]).

```
//////////////////////////////////////// First forest (training on ABC output)
Growing trees ..
  100.0% [ 500/ 500 | 136.2 Hz | 4s<0s]
Computing prediction error ..
//////////////////////////////////////// Second forest (training on error)
Growing trees ..
  100.0% [ 500/ 500 | 555.1 Hz | 1s<0s]
Computing prediction error ..
 votes model1 votes model2 votes model3 votes model4 votes model5 votes model6 selected model post proba
           2           46          410            9           25            8              3      0.741
Predicted model  :  3
votes :  [2, 46, 410, 9, 25, 8]
Posterior probability :  0.7414333333333332
```

Fig. 11. Output of *abcranger* model choice on a pre-generated reftable (as explained before, this is a two stage setup with two forests). The reftable is sampled with the six models, uniformly chosen.

```
Selecting only 9 pls components.
Growing trees ..
                          100.0% [ 500/ 500 | 125.1 Hz | 4s<0s]
Computing prediction error ..

Parameter estimation (point estimates)
   Expectation        Median Quantile_0.05 Quantile_0.95      Variance
      0.387622      0.385098      0.259602      0.534545      0.006349

Global (prior) errors
Computed from the mean taken as point estimate
                     NMAE : 0.17127635307824707
                      MSE : 0.006463761580914472
                     NMSE : 0.02107403354559447
Computed from the median taken as point estimate
                     NMAE : 0.15490037296879525
                      MSE : 0.006874722512837821
                     NMSE : 0.02049269588370233
Confidence interval measures
             90% coverage : 0.9575
              Mean 90% CI : 0.2758619237413165
     Mean relative 90% CI : 0.8453453298782697
           Median 90% CI : 0.23266885779478091
   Median relative 90% CI : 0.5639567421092406

Local (posterior) errors
Computed from the mean taken as point estimate
                     NMAE : 0.1564821612838361
                      MSE : 0.006348615077573065
                     NMSE : 0.015846355369397838
Computed from the median taken as point estimate
                     NMAE : 0.14142338836017088
                      MSE : 0.005765797978743337
                     NMSE : 0.012525174537546181
```

Fig. 12. Output of *abcranger* parameter estimation for *ra* (admixture rate) parameter on DIYABC reftable (12000 samples from the model 3).

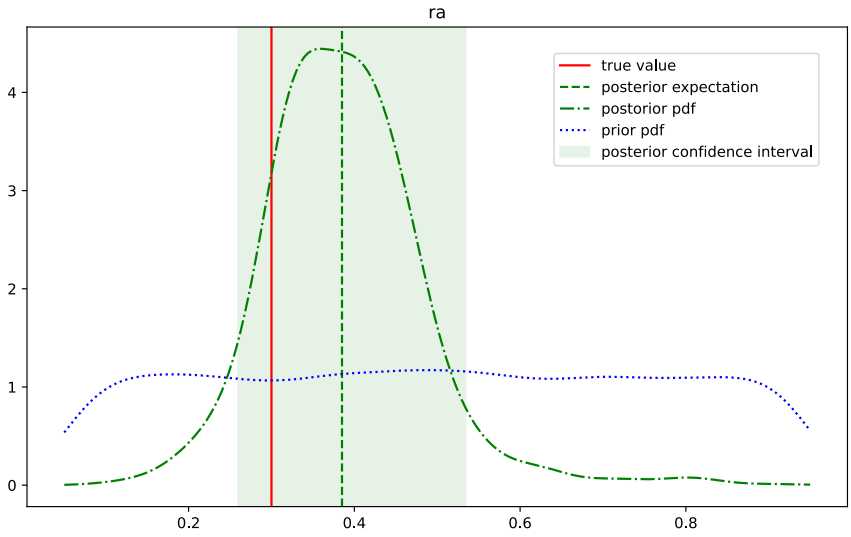

Fig. 13. Inferred posterior distribution of *ra* on model 3

confidence interval. Note that inferring the admixture rate is considered as very difficult problem in population genetics, so the deviation from the real value is expected.

5 Conclusions and perspectives

ABC-RF posterior methodologies are a clean and efficient integration of SML techniques in a model-based approach, although the main objective is not the raw predictive power *per se* like in a pure machine learning perspective, but easy to get, accurate and interpretable posteriors.

Many ideas emphasized in both posterior methodologies from [2] and [3] have strong connections with *Generalized Random Forests* framework [13]. We would like to explore them in order to extend our developments to other fields than population genetics.

Moreover, we intend to pursue the algorithm adaptation of Random Forests for ABC even further, at the tree level: for a growing tree, only encountered leaves should be stored for point estimates and final moments. Thus, the memory footprint of the trees becomes negligible, and their growing could finally be parallelized at full scale.

Finally, by nature of Breiman's *CART*, the computational bottleneck for random forests lies in the greedy, local split procedure at each node. To alleviate this, they are promising optimizations coming from the *Gradient Boosted Trees* community [14] and also some inspired by the *Deep Learning* one like [15].

References

[1] Valentin Hivert, Raphaël Leblois, Eric J Petit, Mathieu Gautier, and Renaud Vitalis. Measuring genetic differentiation from pool-seq data. *Genetics*, 210(1):315–330, 2018.

[2] Pierre Pudlo, Jean-Michel Marin, Arnaud Estoup, Jean-Marie Cornuet, Mathieu Gautier, and Christian P Robert. Reliable ABC model choice via random forests. *Bioinformatics*, 32(6):859–866, 2015.

[3] Louis Raynal, Jean-Michel Marin, Pierre Pudlo, Mathieu Ribatet, Christian P Robert, and Arnaud Estoup. ABC random forests for Bayesian parameter inference. *Bioinformatics*, 35(10):1720–1728, 10 2018.

[4] Jean-Michel Marin, Pierre Pudlo, Christian P Robert, and Robin J Ryder. Approximate bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.

[5] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[6] Leo Breiman, Jerome H Friedman, Charles J Stone, and Richard A Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.

[7] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[8] Jean-Michel Marin, Louis Raynal, Pierre Pudlo, Christian P. Robert, and Arnaud Estoup. *abcrf: Approximate Bayesian Computation via Random Forests*, 2019. R package version 1.8.1.

[9] Nicolai Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7(Jun):983–999, 2006.

[10] Marvin N Wright and Andreas Ziegler. Ranger: a fast implementation of random forests for high dimensional data in c++ and r. *arXiv preprint arXiv:1508.04409*, 2015.

[11] Jarno Lintusaari, Henri Vuollekoski, Antti Kangasrääsiö, Kusti Skytén, Marko Järvenpää, Pekka Marttinen, Michael U. Gutmann, Aki Vehtari, Jukka Corander, and Samuel Kaski. Elfi: Engine for likelihood-free inference. *Journal of Machine Learning Research*, 19(16):1–7, 2018.

[12] Francois David Collin, Ghislain Durif, Louis Raynal, Eric Lombaert, Mathieu Gautier, Renaud Vitalis, Jean Michel Marin, and Arnaud Estoup. Extending approximate bayesian computation with supervised machine learning to infer demographic history from genetic polymorphisms using DIYABC random forest. jul 2020.

[13] Susan Athey, Julie Tibshirani, and Stefan Wager. Generalized random forests. *The Annals of Statistics*, 47(2):1148–1178, Apr 2019.

[14] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, pages 3146–3154, 2017.

[15] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475, 2015.