DIYABC
version 2.1

A user-friendly software for inferring population history through Approximate Bayesian Computations using microsatellite, DNA sequence and SNP data

J.M. Cornuet, P. Pudlo,
J. Veyssier, Etienne Loire, Filipe Santos,
A. Dehne-Garcia and A. Estoup

Centre de Biologie et de Gestion des Populations
Institut National de la Recherche Agronomique
755 avenue du Campus Agropolis, CS 30016
34988 Montferrier-sur-Lez cedex, France
(diyabc@supagro.inra.fr)

July 10, 2015

Contents

1. Preface

1.1 General context and history of DIYABC

In less than 10 years, Approximate Bayesian Computations (ABC) have developed in the Population Genetics community as a new tool for inference on the past history of populations and species. Compared to other approaches based on the computation of the likelihood which are still restrained to a very narrow range of evolutionary scenarios and mutation models, the ABC approach has demonstrated its ability to stick to biological situations that are much more complex and hence realistic. However, this approach still requires numerous computations to be performed so that it has been used mostly by specialists (i.e. statisticians and programmers). This has almost certainly restrained the possible impact of ABC in population genetic studies. We believe that this situation must be improved and therefore we have developed a computer program for the large community of experimental biologists. We therefore designed DIYABC as a user-friendly program allowing non specialist biologists to achieve their own analysis. **The first version (DIYABC v**0.x**)** had been written especially for microsatellite data. There were at least two reasons for that. The first one is that we have been among the first to develop and use this class of markers in population genetic studies (e.g. Estoup *et al.*, 1993). Since then, we have developed microsatellites in numerous species as well as we have published theoretical studies and reviews on these markers (e.g. Estoup *et al.*, 2002). The second reason is that microsatellites have been and still are very popular markers in the population geneticist community and there is now a large quantity of data that might benefit of an ABC approach. **The second version of our software (DIYABC v**1.x**)** has been designed to make use of DNA sequence data. This has several immediate consequences. For instance, the standard Genepop data file format has been extended to incorporate sequence data. This has been done in collaboration with the authors of *Genepop* and explained in subsection 4.1.1. In this version, sequence loci are considered in the same way as microsatellite loci, i.e. they are considered as genetically independent and intra-locus recombination is not (yet) available. Regarding mutation models for DNA sequences, we used the same philosophy as for microsatellites, i.e. the program considers only simple and widely used models, keeping in mind that a higher-dimensional parameter space will be less well explored than a lower-dimensional space. Note that none of these mutation models includes insertion-deletions. Also five categories of loci (either microsatellites or DNA sequences) were considered in this second version : autosomal diploid, autosomal haploid, X-linked, Y-linked and mitochondrial. Note that X-linked loci can be used for an haplo-diploid species in which both sexes have been sampled. If non-autosomal loci have been typed in population samples, the sex-ratio of the species will have to be provided (see subsection 4.1.1).

Other improvements over version 0 included :

1. the use of multithread technology in order to exploit multicore/multiprocessor computers. This is especially useful when building the reference table and for several other intensive computation steps, such as the multinomial logistic regression,
2. a new option which helps the detection of "bad" prior modelisation of the data,
3. another new option which helps evaluate the goodness of fit of a given model-parameter posterior combination (i.e. Model checking),
4. many new screens implemented not only to treat sequence data, but also to cope with the new options described above, as well as to offer useful complementary information on the current run.

The third version of DIYABC (DIYABC v2.x**)** has been entirely recoded in order to be used under the usual three OS (Linux, Windows and Mac). Also the code for computations has been separated from that of the graphic user interface (GUI). The former has been rewritten in C++ and the latter is a mixture of Python and Qt (PyQt). The user can then launch computations with or without using the GUI. The GUI 's uses are :

1. the management of projects
2. the input of the historical and genetical models
3. the parameterization of analyses
4. the launch of computations of the reference table and of the various required analyses

5. the visualization of results

Also, as DNA sequences have been added in the second version, a new category of markers has been added to the third version : Single Nucleotide Polymorphisms (SNPs). Instead of extending once more the Genepop format, a new data simple format has been designed for these markers. Note that SNP data are treated separately from other markers (*i.e.* they cannot be analyzed together with microsatellite and/or DNA sequence data). It is worth noting that, in the present version of the program, the analysed SNP data are assumed to correspond to independent selectively-neutral loci, without any ascertainment bias (i.e. the deviations from expected theoretical results due to the SNP discovery process in which a small number of individuals from selected populations are used as discovery panel).

This version includes all improvements of version 1.x and a few new improvements such as :

- loci of the same type (i.e. microsatellites on one hand or DNA sequences on the other hand) can be associated in one or more groups. This allows for instance to define different mutation models for microsatellites with motifs of different lengths.
- the model checking option is now presented as a direct option (not a suboption of the ABC estimation of parameters) which largely simplifies its use.
- the logistic regression can be performed on linear discriminant analysis components instead of all summary statistics. This reduces the number of dependent variables, thus allowing to run large "confidence in scenario choice" analyses including many summary statistics and scenarios (Estoup *et al.*, 2012).

The latest version of the program (DIYABC v2.1.0) includes the following major improvements: (i) new analysis options to compute error / accuracy indicators conditionally to the observed dataset, (ii) possibility to specify a MAF (minimum allele frequency) criterion on the analyzed SNP datasets, and (iii) optimization of the simulation process of SNP datasets that include a substantial amount of missing data.

1. New analysis options to compute error / accuracy indicators conditionally to the observed dataset.

The program DIYABC allows evaluating the confidence in scenario choice and the accuracy of parameter estimation under a given scenario using simulated pseudo-observed datasets (pods), for which the true scenario ID and parameter values are known. So far such pods were drawn randomly into prior distributions for both the scenario ID and the parameter values. By doing so, we estimate global error/accuracy levels computed over the whole (and usually huge) data space defined by the prior distributions. These indicators hence actually correspond to "prior" error rates (when evaluating the confidence in scenario choice) or "prior" precision measures (when evaluating the accuracy of parameter estimation under a given scenario). The levels of error/accuracy may be substantially different depending on the location of an observed or pseudo-observed dataset in the prior data space. Indeed, some peculiar combination of parameter values may correspond to situations of strong (weak) discrimination among the compared scenarios or of accurate (inaccurate) estimation of parameter values under a given model. Aside from their use to select the best classifier and set of summary statistics, prior-based indicators are, however, poorly relevant since, for a given dataset, the only point of importance in the data space is the observed dataset itself. Computing error / accuracy indicators conditionally to the observed dataset (i.e. focusing around the observed dataset by using the posterior distributions) is hence clearly more relevant than blindly computing indicators over the whole prior data space as done so far. This is basically what DIYABC v2.1.0 proposes to do with several new analysis sub-options available within the options "Evaluate confidence in the scenario choice" and "Compute bias and precision on parameter estimations". Indeed, one can now choose to compute a "posterior" error rate (when evaluating the confidence in scenario choice) by drawing the scenario ID and parameter values of a large number of pods from the s simulated datasets closest to the observed dataset (i.e. the s datasets with the smallest Euclidean distance). Typically, $s = 500$ (when simulating 10,000 to 1 million datasets per compared scenario) but this number can be lowered to 100. In the same vein, one can now choose to compute "posterior" accuracy indicators (when evaluating the accuracy of parameter estimation under a given scenario) by drawing the parameter values of a large number of pods among the parameter posterior distributions estimated under a given scenario using a standard ABC procedure. Note that we found, using controlled genetics experiments, that posterior error (accuracy) measures could strongly differ from prior error (accuracy) measures, hence making a case of the significance of computing error (accuracy) measures conditionally to the observed dataset rather than blindly computing such measures over the whole prior data space (unpublished results and see Pudlo et al. 2015).

2. Possibility to specify a MAF (minimum allele frequency) criterion on the analyzed SNP datasets.

Compared to other types of molecular markers, SNP loci have low mutation rates, so that polymorphism at such loci results from a single mutation during the whole population(s) gene tree and genotypes are bi-allelic. To generate a simulated polymorphic dataset at a given SNP locus, we proceeded following the algorithm proposed by Hudson (2002) (cf –s 1 option in the program ms associated to Hudson, 2002). Briefly, the genealogy at a given locus of all genes sampled in all populations of the studied dataset is simulated until the most recent common ancestor according to coalescence theory. Then a single mutation event is put at random on one branch of the genealogy (the branch being chosen with a probability proportional to its length relatively to the total gene tree length). This algorithm provides the simulation efficiency and speed necessary in the context of ABC, where large numbers of simulated datasets including numerous SNP loci have to be generated (Cornuet et al. 2014). Most importantly, using the Hudson's simulation algorithm is equivalent to applying a default MAF (minimum allele frequency) criterion on the simulated dataset. As a matter of fact, each locus in both the observed and simulated datasets will be characterized by the presence of at least a single copy of a variant over all genes sampled from all studied populations (i.e. pooling all genes genotyped at the locus). In DIYABC v2.1.0, it is possible to impose a different MAF criterion for each locus on the observed and simulated datasets. This MAF is computed pooling all genes genotyped over all studied population samples. For instance, the specification of a MAF equal to 5% will automatically select a subset of m loci characterized by a minimum allele frequency > 5% among the l locus of the observed dataset. In agreement with this, only m locus with a MAF>5% will be retained in a simulated dataset (simulated loci with a MAF≤5% will be discarded). In practice, the instruction for a given MAF has to be indicated directly in the headline of the observed dataset. For instance, if one wants to consider only loci with a MAF equal to 5% one will write <MAF=0.05> in the headline. Writing <MAF=hudson> (or omitting to write any instruction with respect to the MAF) will bring the program to use the standard Hudson's algorithm without further selection as done so far in the previous version of DIYABC. The selection with DIYABC v2.1.0 of a subset of loci fitting a given MAF allows: (i) to remove the loci with very low level of polymorphism from the dataset and hence increase the mean level of genetic variation of both the observed and simulated datasets, without producing any bias in the analyses; and (ii) to reduce the proportion of loci for which the observed variation may corresponds to sequencing errors. In practice MAF values ≤10% are considered. To check for the consistency/robustness of the ABC results obtained, it may be useful to treat a SNP dataset considering different MAFs (for instance MAF=hudson, MAF=0.01 and MAF=0.05).

3. Optimization of the simulation process of SNP datasets that include a substantial amount of missing data.

We have radically changed our way to take into account missing data for SNP datasets (i.e. missing genotypes denoted "9" in the data file). The initial way to deal with missing data turned out to be poorly efficient in term of computation time, especially when the number of SNP missing data was large which seems to be the case for many real SNP datasets. The new code we have implemented to deal with this issue is particularly efficient and makes it feasible to simulate in a reasonable time large SNP datasets including (or not) numerous missing data.

Finally, as for DIYABC v1, the most recent versions of DIYABC v2 (v2.0 and v2.1) deals with sexually reproducing diploid or haploid species (co-dominant markers corresponding to autosomal, X-linked, Y-linked loci) but does not allow considering species reproducing clonally.

For all versions of DIYABC, we recommend non-expert users to use the GUI for their computations.

1.2 References to cite

- **version 0 :** Cornuet J.M., F. Santos, M.A. Beaumont, C.P. Robert, J.M. Marin, D.J. Balding, T. Guillemaud and A. Estoup. 2008. Inferring population history with DIYABC: a user-friendly approach to Approximate Bayesian Computations. *Bioinformatics*, **24** (23), 2713-2719.

- **version 1 :** Cornuet J.M., V. Ravignï¿œ and A. Estoup, 2010. Inference on population history and model checking using DNA sequence and microsatellite data with the sofware DIYABC (v1.0) (2010) *BMC Bioinformatics*, **11**, 401.

- **version 2 :** Cornuet, J-M., Pudlo, P., Veyssier, J., Dehne-Garcia A., Gautier M., Leblois R., Marin J-M, and A. Estoup, 2014. DIYABC v2.0: a software to make approximate Bayesian computation inferences about population history using single nucleotide polymorphism, DNA sequence and microsatellite data. *Bioinformatics*. Vol. 30, no. 8, p1187–1189, doi: 10.1093/bioinformatics/btt763.

1.3 Web site

http://www1.montpellier.inra.fr/CBGP/DIYABC
You can get there executable files for different operating systems as well as the last version of this detailed notice document.

1.4 System requirements

- DIYABC should run on any linux flavour, Microsoft Windows XP and Seven and OS x 10.5 (intel) or later.
- Minimum 4GB of RAM; 6GB of RAM recommended
- 70MB free disk space for DIYABC binaries
- From 1 to 10GB free disk space for each project depending on the project configuration and the records number in the reftable file.

Caveat : it is possible that on windows 32bits (and sometimes on windows 64bits) the reference table file will not grow more than 4Go. We hope to be able to circumvent this constraint soon on windows 64bits.

1.5 How to create (and send) a bug report

DIYABC V2 provides an easy way to send to the program developers the different files and clues that are necessary to attempt solving a bug. Click on the âHelpâ menu. Then go on the âCreate bug reportâ tab. You then just need to following the few instructions we give you at this stage, validate and save the created bug report tarred file somewhere on your computer. Please send the bug report file to the indicated email address (`DIYABC@supagro.inra.fr`). Two remarks here: (i) the bug you describe has to be the last things that you did with the program; and (ii) please try to reproduce your bug one time and then create and send the bug report. Finally, it had to be noted that if the bug completely crash the application then no bug report can be created. We will do our best to solve your bug thanks to the bug report you provided us.

1.6 Acknowledgements

We thank Mark Beaumont who has been at the origin of our interest for ABC. He offered us constant help and inspiration since the beginning. We also thank David Balding who welcomed one of us (JMC) in his team during the whole writing of the program and who organized several workshops on ABC during the same period. We are indebted to Christian Robert, Jean-Michel Marin, Stuart Baird, Thomas Guillemaud, Renaud Vitalis, Gael Kergoat, Gilles Guillot and David Welsh with whom we discussed many theoretical and practical aspect of DIYABC in the numerous meetings financed by a grant from the French Research National Agency (project *MISGEPOP* ANR-05-BLAN-196). The same grant is also aknowledged for having paid for the 2-year salary of FS. This research was also supported by an EU grant awarded to JMC as an EIF Marie-Curie fellowship (project *StatInfPopGen*) and which allowed him to come to David Balding's place at Imperial College (London, UK). Current and future developments of DIYABC are financed by a new grant from the French Research National Agency (project *EMILE* ANR-09-BLAN-0145) awarded in september 2009. We thank several âbeta-usersâ, especially Eric Lombaert, Michael Fontaine, Christophe Plantamp, Marie-Pierre Chapuis, Carine Brouat, RaphaÃl Leblois and Thomas Guillemaud, who tested the the DIYABC V2 software with their data.

2. Methodology

2.1 Basic notions on ABC

Approximate Bayesian Computation or ABC is a bayesian approach in which the posterior distributions of the model parameters are determined by replacing the computation of the likelihood (probability of observed data given the values of the model parameters) by a measure of similarity between observed and simulated data. The posterior distributions are estimated from parameter values providing simulated data that are the most similar to observed data. Historically, different ways of estimating this similarity have been proposed, but all have been based on statistics summarizing information conveyed by the data set. In population genetics, data most often relate to individuals that have been genotyped at a given set of loci, these individuals being representative of the studied populations. The summary statistics are for instance the mean number of alleles per population or genetic distances between pairs of populations. It is much easier to measure the similarity between small sets of summary statistics than between large sets of multilocus genotype data. When the number of summary statistics is low, it is possible to select simulated data for which *all* the summary statistics are close to those of the observed data (Pritchard *et al.*, 1999; Estoup *et al.*, 2001; Estoup and Clegg, 2003). However, for more complex scenarios necessitating a larger number of summary statistics, it becomes almost impossible to find such simulated data sets. Beaumont *et al.* (2002) have hence proposed to measure similarity through the Euclidian distance between observed and simulated summary statistics, after normalization by standard deviations of simulated statistics. In addition, these authors introduced a step of weighted local linear regression aimed at favoring simulated data sets that are closer to the observed one.
In practice, the ABC approach can be summarized in three successive steps (Excoffier *et al.*, 2005) : i) generating simulated data sets, ii) selecting simulated data sets closest to observed data set and iii) estimating posterior distributions of parameters through a local linear regression procedure.
In addition, this approach provides a way of comparing different models (hereafter named scenarios) that can explain observed data. Two measures of posterior probabilities of scenarios are proposed. The first measure is simply the relative proportion of each scenario in the simulated data sets closest to observed data sets (Miller *et al.*, 2005; Pascual *et al.*, 2007). The second measure is obtained by a logistic regression of each scenario probability on the deviations between simulated and observed summary statistics (Fagundes *et al.*, 2007; Beaumont, 2008).

In order to simulate data, one has first to define one (or possibly several) scenario(s). Each scenario includes a historical model describing how the sampled populations are connected to their common ancestor and a mutational model describing how allelic states of the studied genes are changing along their genealogical trees.

2.2 Historical model parameterization

The evolutionary scenario, which is characterized by the historical model, can be described as a succession in time of "events" and "inter event periods". The events considered in the program are a restricted set of possible events affecting populations evolution. In the current version of the program, we consider only 4 categories of events : population divergence, discrete change of effective population size, admixture and sampling (the last one has been added to allow considering samples taken at different times). Between two successive events affecting a population, we assume that populations evolve independently (e.g. without migration) and with a fixed effective size. The usual parameters of the historical model are the times of occurrence of the various events (counted in generations), the effective sizes of populations and the admixture rates. When writing the scenario, events have to be coded sequentially backward in time (see section *2.5 Prior Distribution* when time priors are overlapping). Although this choice may not be natural at first sight, it is coherent with coalescence theory on which are based all data simulations in the program. For that reason, the keywords for a divergence or an admixture event are `merge` and `split`, respectively. Two other keywords, `varNe` and `sample`, correspond to a discrete change in effective population size and a gene sampling, respectively.
A scenario takes the form of a succession of lines (one line per event), each line starting with the time of the event, then the nature of the event, and ending with several other data depending on the nature of the event. Following is the syntax used for each category of event :

population sample : ⟨*time*⟩ `sample` ⟨*pop*⟩ ⟨*time*⟩ is the time (always counted in number of generations) at which the sample was taken and

$\langle pop \rangle$ is the population number from which is taken the sample. It is worth stressing here that **samples are considered in the same order as they appear in the data file**.

population size variation : $\langle time \rangle$ `varNe` $\langle pop \rangle$ $\langle Ne \rangle$
From time $\langle time \rangle$, looking backward in time, population $\langle pop \rangle$ will have an effective size $\langle Ne \rangle$.

population divergence : $\langle time \rangle$ `merge` $\langle pop0 \rangle$ $\langle pop1 \rangle$
At time $\langle time \rangle$, looking backward in time, population $\langle pop1 \rangle$ "merges" with population $\langle pop0 \rangle$. Hereafter, only $\langle pop0 \rangle$ "survives".

population admixture : $\langle time \rangle$ `split` $\langle pop0 \rangle$ $\langle pop1 \rangle$ $\langle pop2 \rangle$ $\langle rate \rangle$
At time $\langle time \rangle$, looking backward in time, population $\langle pop0 \rangle$ "splits" between populations $\langle pop1 \rangle$ and $\langle pop2 \rangle$. A gene lineage from population $\langle pop0 \rangle$ joins population $\langle pop1 \rangle$ (respectively $\langle pop2 \rangle$) with probability $\langle rate \rangle$ (respectively 1-$\langle rate \rangle$).

A historical model is a succession of lines as described above. However, in order to cope with special situations (see explanations in Note 9 below), we added a first line giving the effective sizes of sampled populations before the first event described, looking backward in time. Expressions between arrows, other than population numbers, can be either a numeric value (e.g. 25) or a character string (e.g. `t0`). In the latter case, it is considered as a parameter of the model. So the only possible parameters of the historical model are times of events, effective population sizes and admixture rates.
The program offers the possibility to add or remove scenarios, by just clicking on the corresponding buttons. The usual shortcuts (CTRL+C, CTRL+V and CTRL+X) can be used to edit the different scenarios. Some or all parameters can be in common among scenarios.

Notes

1. There are two ways of giving a fixed value to effective population sizes, times and admixture rates. Either the fixed value appears as a numeric value in the scenario windows or it is given as a string value like any parameter. In the latter case, one gives this parameter a fixed value by choosing a Unifom distribution and setting the minimum and maximum to that value in the prior setting (see section 2.4).

2. All expressions must be separated by at least one space.

3. All expressions relative to parameters can include sums or differences. For instance, it is possible to write :
   `t0 merge 2 3`
   `t0+t1 merge 1 2`
   This means that `t1` is the time elapsed between the two events. By imposing `t1>0` (as explained in section **prior and posterior distributions**), this implies that the divergence of populations 1 and 2 is always more ancient than the divergence of populations 2 and 3. However, one cannot mix a parameter and a numeric value (e.g. `t1+150` will result in an error). This can be done by writing `t1+t2` and fixing `t2` by choosing a uniform distribution with lower and upper bounds both equal to 150.

4. Time is always given in generations. Since we look backward, time increases towards past.

5. Negative times are allowed (e.g. the example given in section 3), but not recommended.

6. Population numbers must be consecutive natural integers starting at 1. The number of population can exceed the number of samples and vice versa : in other words, unsampled populations can be considered in the scenario on one hand, and the same population can be sampled more than once on the other hand.

7. Multi-furcating population trees can be considered, by writing several divergence events occurring at the same time. However, one has to be careful to the order of the `merge` events. For instance, the following piece of scenario will fail :
   `100 merge 1 2`
   `100 merge 2 3`
   This is because, after the first line, population 2, which has merged with population 1, does not

"exist" anymore (the surviving population is population 1). So, it cannot receive lineages of population 3 as it should as a result of the second line. The correct ways are either to put line 2 before line 1, or to change line 2 to :
`100 merge 1 3`.

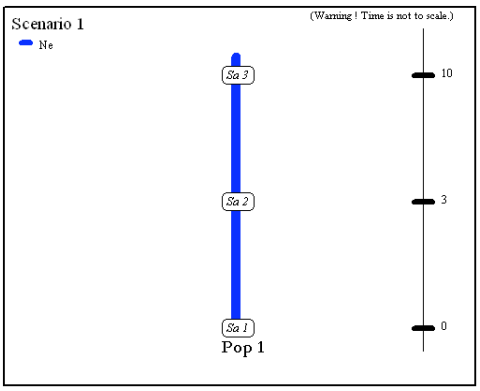
8. Since times of events can be parameters, the order of events can change according to the values taken by the time parameters. In any case, before simulating a data set, the program sorts out events by increasing times [1]. If two or more events occur at the same time, the order is that of the scenario as it is written by the the user.

9. Most scenarios begin with sampling events. We then need to know the effective size of the populations to perform the simulation of coalescences until the next event concerning each population. One way would have been to provide the population size on the same line of the scenario description. However, in some scenarios with varying population sizes, it can not be determined what is the effective size at the sampling time before the set of time parameter values is generated. For that reason, we decided to provide the effective size and the sampling description on two distinct lines.

Examples Below are some usual scenarios with increasing complexity. Each scenario is coded on the left side and a graphic representation given by DIYABC is printed on the right side

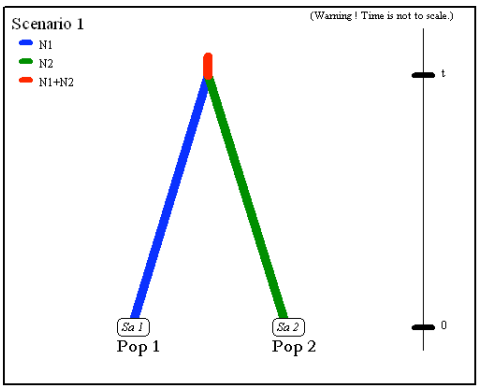
1. One population from which several samples have been taken at various generations : 0, 3 and 10. The only unknown parameter of the scenario[2] is the effective population size.

```
Ne
0 sample 1
3 sample 1
10 sample 1
```



2. Two populations of size `N1` and `N2` have diverged `t` generations in the past from an ancestral population of size `N1+N2`.

```
N1 N2
0 sample 1
0 sample 2
t merge 1 2
t varNe 1 N1+N2
```

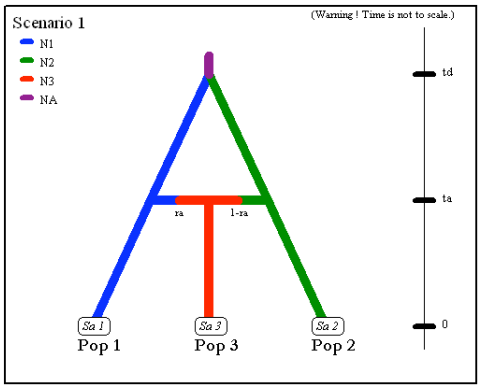


[1] Sorting events by increasing times can only be done when all time values are known, i.e. when simulating datasets. When checking scenarios, all time values are not yet defined, so that when visualizing a scenario, events are represented in the same order as they appear in the window used to define the scenario.

[2] Of course, there are also one or more parameter(s) for the mutation model.

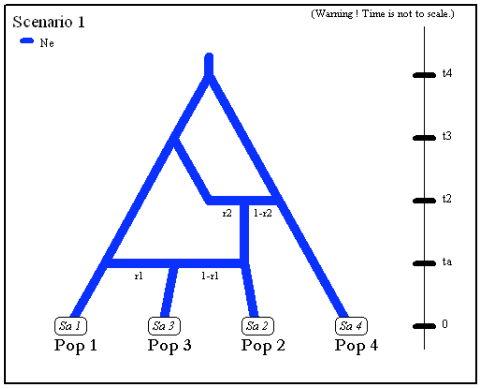
3. Two parental populations (1 and 2) with constant effective populations sizes `N1` and `N2` have diverged at time `td` from an ancestral population of size `NA`. At time `ta`, there has been an admixture event between the two populations giving birth to an admixed population (3) with effective size `N3` and with an admixture rate `ra` relative to population 1.

```
N1 N2 N3
0 sample 1
0 sample 2
0 sample 3
ta split 3 1 2 ra
td merge 1 2
td varNe 1 NA
```



4. The next scenario is slightly more complicated. It includes four population samples and two admixture events. For simplicity sake, all populations are assumed to have identical effective sizes (`Ne`).

```
Ne Ne Ne Ne Ne
0 sample 1
0 sample 2
0 sample 3
0 sample 4
t1 split 3 1 2 r1
t2 split 2 5 4 r2
```

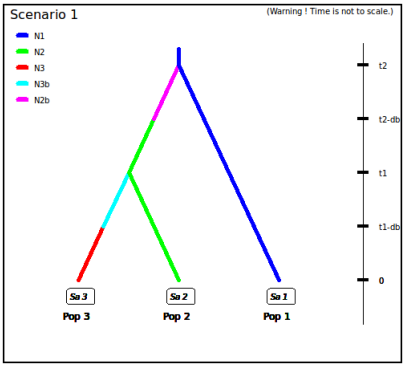


Note that although there are only four samples, the scenario includes a fifth unsampled population. This unsampled population which diverged from population 1 at time `t3` was a parent in the admixture event occurring at time `t2`. Note also that the first line must include the effective sizes of the *five* populations.

5. The following three scenarii correspond to a classic invasion history from an ancestral population (population 1). In scenario 1, population 3 is derived from population 2, itself derived from population 1. In scenario 2, population 2 derived from population 3, itself derived from population 1. In scenario 3, both populations 2 and 3 derived independently from population 1. The same trio of scenarii will be taken later in a fully described example. Note that when a new population is created from its ancestral population, there is an initial size reduction (noted here `N2b` for population 2 and `N3b` for population 3) since the invasive population generally starts with a few immigrants.

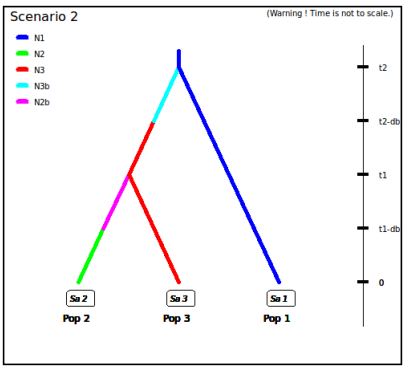
Scenario 1

```
   N1 N2 N3
0 sample 1
0 sample 2
0 sample 3
t1-db VarNe 3 N3b
t1 merge 2 3
t2-db VarNe 2 N2b
t2 merge 1 2
```



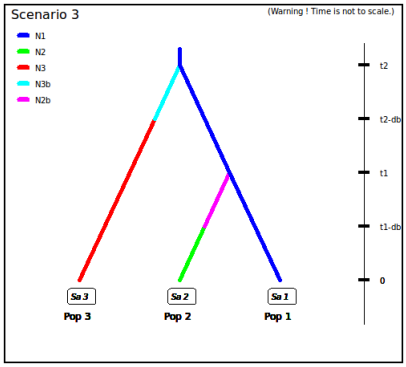
Scenario 2

```
   N1 N2 N3
0 sample 1
0 sample 2
0 sample 3
t1-db VarNe 2 N2b
t1 merge 3 2
t2-db VarNe 3 N3b
t2 merge 1 3
```



Scenario 3

```
   N1 N2 N3
0 sample 1
0 sample 2
0 sample 3
t1-db VarNe 2 N2b
t1 merge 1 2
t2-db VarNe 3 N3b
t2 merge 1 3
```



2.3 Mutation model parameterization (microsatellite and DNA sequence loci)

The program can analyse microsatellite data and DNA sequence data altogether as well as separately. In the current version, there are still two restrictions. First, all loci in an analysis must be genetically independent. Second, for DNA sequence loci, intralocus recombination is not considered.

Loci are grouped by the user according to its needs (this an improvement of the current version which imposed all loci of a given category to follow the same mutation model). A different mutation model can be defined for each group. For instance, one group can include all microsatellites with motifs that are 2 bp long and another group those with a 4 bp long motif. Also, with DNA sequence loci, nuclear loci can be grouped together and a mitochondrial locus form a separate group.

The parameterization of the two categories of markers is now described below.

2.3.1 Microsatellite loci

Although a variety of mutation models have been proposed for microsatellite loci (Whittaker *et al.*, 2003), it is usually sufficient to consider only the simplest models (Cornuet *et al.*, 2006). This has the non-negligible advantage of reducing the number of parameters, which can be a real issue when complex scenarios are considered. This is why we chose the Generalized Stepwise Mutation model (Estoup *et al.*, 2002). Under this model, a mutation increases or decreases the length of the microsatellite by a number of repeated motifs following a geometric distribution. This model necessitates only two parameters : the mutation rate (μ) and the parameter of the geometric distribution (P). The same mutation model is imposed to all loci of a given group. However, each locus has its own parameters (μ_i and P_i) and, following a hierarchical scheme, each locus parameter is drawn from a gamma distribution with mean equal to the mean parameter value. Note also that :

1. individual loci parameters (μ_i and P_i) are considered as nuisance parameters and hence are never recorded. Only mean parameters are recorded.
2. The variance or shape parameter of the gamma distributions are set by the user and are NOT considered as parameters.
3. The SMM or Stepwise Mutation Model is a special case of the GSM in which the number of repeats involved in a mutation is always one. Such a model can be easily achieved by setting the maximum value of mean P ($\bar{P}$) to 0. In this case, all loci have their P_i set equal to 0 whatever the shape of the gamma distribution.
4. All loci can be given the same value of a parameter by setting the shape of the corresponding gamma distribution to 0 (this is NOT a limiting case of the gamma, but only a way of telling the program).

Eventually, to give more flexibility to the mutation model, the program offers the possibility to consider mutations that insert or delete a single nucleotide to the microsatellite sequence. In the previous version, this option was considered as marginal, and was not treated in the same way as the motif size stepwise mutational process, i.e. there was no associated parameter that could be adjusted to the data. This has been changed in this version : it is now possible to use a mean parameter (named $\mu_{(SNI)}$) with a prior to be defined and individual loci having either values identical to the mean parameter or drawn from a Gamma distribution.

2.3.2 DNA sequence loci

Note first that this version of the program does not consider insertion-deletion mutations, mainly because there does not seem to be much consensus on this topic. Concerning substitutions, only the simplest models are considered. We chose the Jukes-Cantor (1969) one parameter model, the Kimura (1980) two parameter model, the Hasegawa-Kishino-Yano (1985) and the Tamura-Nei (1993) models. The last two models include the ratios of each nucleotide as parameters. However, in order to reduce the number of parameters, these ratios have been fixed to the values calculated from the observed data set for each DNA sequence locus. Consequently, this leaves two and three parameters for the Hasegawa-Kishino-Yano (HKY) and Tamura-Nei (TN), respectively. Also, two adjustments are possible : one can fix the fraction of constant sites (those that cannot mutate) on the one hand and the shape of the Gamma distribution

of mutations among sites on the other hand.
As for microsatellites, all sequence loci of the same group are given the same mutation model with mean parameter(s) drawn from priors and each locus has its own parameter(s) drawn from a Gamma distribution (same hierarchical scheme). Notes 1, 2 and 4 of previous subsection (2.3.1) apply also for sequence loci.

2.4 SNPs do not require mutation model parameterization

SNPs have two characteristics that allow to get rid of mutation models : they are polymorphic and they present only two allelic (ancestral and derived) states. In order to be sure that all analyzed SNP loci have the two characteristics, non polymorphic loci are disgarded right from the beginning of analyses. Note that a warning message will appear if the observed dataset include monomorphic loci, the latter being automatically removed from further analyses by the program. Consequently, no matter *how* it occurred, we can assume that there occured one and only one mutation in the coalescence tree of sampled genes. We will see below that this largely simplifies (and speeds up) SNP data simulation as one can use in this case the efficient algorithm of Hudson (2002) (Cornuet et al. 2014). Also, this advantageously reduces the dimension of the parameter space (as mutation parameters are not needed in this case). There is however a potential drawback which is the absence of any calibration generally brought by priors on mutation parameters. Consequently, (time/effective size) ratios rather than original time parameters will be informative.

It is worth stressing that, using the Hudson's simulation algorithm for SNP markers is equivalent to appling a default MAF (minimum allele frequency) criterion on the simulated dataset. As a matter of fact, each locus in both the observed and simulated datasets will be characterized by the presence of at least a single copy of a variant over all genes sampled from all studied populations (i.e. pooling all genes genotyped at the locus). In DIYABC v2.1.0, it is possible to impose a different MAF criterion for each locus on the observed and simulated datasets. This MAF is computed pooling all genes genotyped over all studied population samples. For instance, the specification of a MAF equal to 5% will automatically select a subset of m loci characterized by a minimum allele frequency $> 5\%$ among the l locus of the observed dataset. In agreement with this, only m locus with a MAF>5% will be retained in a simulated dataset (simulated loci with a MAF$\leq$5% will be discarded). In practice, the instruction for a given MAF has to be indicated directly in the headline of the observed dataset. For instance, if one wants to consider only loci with a MAF equal to 5% one will write <MAF=0.05> in the headline. Writing <MAF=hudson> (or omitting to write any instruction with respect to the MAF) will bring the program to use the standard Hudson's algorithm without further selection as done so far in the previous version of DIYABC. The selection with DIYABC v2.1.0 of a subset of loci fitting a given MAF allows: (i) to remove the loci with very low level of polymorphism from the dataset and hence increase the mean level of genetic variation of both the observed and simulated datasets, without producing any bias in the analyses; and (ii) to reduce the proportion of loci for which the observed variation may corresponds to sequencing errors. In practice MAF values $\leq$10% are considered. To check for the consistency/robustness of the ABC results obtained, it may be useful to treat a SNP dataset considering different MAFs (for instance MAF=hudson, MAF=1% and MAF=5%).

2.5 Prior distributions

The Bayesian aspect of the ABC approach implies that parameter estimations use prior knowledge about these parameters, prior knowledge given by prior distributions of parameters. The program offers a choice among usual probability distributions, i.e. Uniform, Log-Uniform, Normal or Log-Normal for historical parameters and Uniform, Log-Uniform or Gamma for mutation parameters. Extremum values and other parameters (e. g. mean and standard deviation) must be filled in by the user.
In addition, one can impose some simple conditions on historical parameters. For instance, there can be two times parameters with overlapping prior distributions. However, we want that the first one, say `t1`, to always be larger than the second one, say `t2`. For that, we just need to set `t1` > `t2` in the corresponding edit-windows. Such a condition needs to be between two parameters (not a parameter and a number, though this can be set up by giving a minimum and a maximum to the prior distribution) and more precisely between two parameters of the same category (i.e. two effective sizes, two times or two admixture rates). The limit to the number of conditions is imposed by the logics, not by the program. The only binary relationships accepted here are $>, <, >= and <=$.

2.6 Algorithms for data simulation : main features

Data simulation is based on the Wright-Fisher model. It consists in generating the genealogy of all sampled genes until their most recent common ancestor using coalescence theory.
This begins by randomly drawing a complete set of parameters from their own prior distributions and that satisfy all imposed conditions. Then, once events have been ordered by increasing times, a sequence of *actions* is constructed. If there are more than one locus, the same sequence of actions is used for all successive loci. Possible *actions* fall into four categories :

adding a sample to a population :
Add as many gene lineages to the population as there are genes in the sample.

merge two populations :
Move the lineages of the second population into the first population.

split between two populations :
Distribute the lineages of the admixed population among the two parental populations according to the admixture rate.

coalesce and mutate lineages within a population :
There are two possibilities here, depending on whether the population is *terminal* or not. We call *terminal* the population including the most recent common ancestor of the whole genealogy. In a terminal population, coalescences and mutations stop when the MRCA is reached whereas in a non terminal population, coalescence and mutations stop when the upper (most ancient) limit is reached. In the latter case, coalescences can stop before the upper limit is reached because there remains a single lineage, but this single remaining lineage can still mutate.
Two different algorithms are implemented : a generation by generation simulation or a continuous time simulation. The choice, automatically performed by the program, is based on an empirical criterion which ensures that the (approximate[3]) continuous time algorithm is chosen whenever it is faster than the (exact[3]) generation by generation while keeping the relative error on the coalescence rate below 5% (see Cornuet *et al.* (2008) for a description of this criterion).
In any case, a coalescent tree is generated over all sampled genes.
Then the simulation process diverges depending on the type of markers : for microsatellite or DNA sequence loci, mutations are distributed over the branches according to a Poisson process whereas for SNP loci, one mutation is applied to a single branch of the coalescent tree, this branch being drawn at random with probability proportional to its length.
Eventually, starting from an ancestral allelic state (established as explained below), all allelic states of the genealogy are deduced forward in time according to the mutation process. For microsatellite loci, the ancestral allelic state is taken at random in the stationary distribution of the mutation model (not considering potential single nucleotide indel mutations). For DNA sequence loci, the procedure is slightly more complicated. First, the total number of mutations over the entire tree is evaluated. Then according to the proportion of constant sites and the gamma distribution of individual site mutation rates, the number and position of mutated sites are generated. Finally, these mutated sites are given 'A', 'T', 'G' or 'C' states according to the selected mutation model. For SNP loci, the ancestral allelic state is arbitrarily set to 0 and it becomes equal to 1 after le the mutation.
Each category of loci has its own coalescence rate deduced from male and female effective population sizes . In order to combine different categories (e.g. autosomal and mitochondrial), we have to take into account the relationships among the corresponding effective population sizes. This can be achieved by linking the different effective population sizes to the effective number of males (N_M) and females (N_F) through the sum $N_T = N_F + N_M$ and the ratio $r = N_M/(N_F + N_M)$. We use the following formulae for the probability of coalescence of two lineages within this population :

autosomal diploid loci : $p = \frac{1}{8r(1-r)N_T}$

autosomal haploid loci : $p = \frac{1}{4r(1-r)N_T}$

X-linked loci / haplo-diploid loci : $p = \frac{1+r}{9r(1-r)N_T}$

Y-linked loci : $p = \frac{1}{rN_T}$

[3]The terms *approximate* and *exact* are relative to the basic assumptions of the Wright-Fisher model, not to the biological reality of the process.

Mitochondrial loci : $p = \frac{1}{(1-r)N_T}$

Users have to provide a (total) effective size N_T (on which inferences will be made) and a sex-ratio r. If no sex ratio is provided, the default value of r is taken as 0.5.

2.7 Summary statistics

For each category (microsatellite, DNA sequences or SNP) of loci, the program proposes a series of summary statistics among those used by population geneticists. These summary statistics are mean values or variances over loci of the same group and characterize a single, a pair or a trio of population samples. These are :

2.7.1 for microsatellite loci

Single sample statistics :

1. mean number of alleles across loci
2. mean gene diversity across loci (Nei, 1987)
3. mean allele size variance across loci
4. mean M index across loci (Garza and Williamson, 2001; Excoffier *et al.*, 2005)

Two sample statistics :

1. mean number of alleles across loci (two samples)
2. mean gene diversity across loci (two samples)
3. mean allele size variance across loci (two samples)
4. F_{ST} between two samples (Weir and Cockerham, 1984)
5. mean index of classification (two samples) (Rannala and Moutain, 1997; Pascual *et al.*, 2007)
6. shared allele distance between two samples (Chakraborty and Jin, 1993)
7. $(\delta\mu)^2$ distance between two samples (Golstein *et al.*, 1995)

Three sample statistics :

1. Maximum likelihood coefficient of admixture (Choisy *et al.*, 2004)

2.7.2 for DNA sequence loci

Single sample statistics :

1. number of distinct haplotypes
2. number of segregating sites
3. mean pairwise difference
4. variance of the number of pairwise differences
5. Tajima's D statistics (Tajima, 1989)
6. Number of private segregating sites (=number of segregating sites if there is only one sample)
7. Mean of the numbers of the rarest nucleotide at segregating sites[4]
8. Variance of the numbers of the rarest nucleotide at segregating sites

Two sample statistics :

1. number of distinct haplotypes in the pooled sample
2. number of segregating sites in the pooled sample

[4] This statistics can provide information in case of recent demographic variation : a recent expansion increases the number of singletons (nucleotides occuring just once at a segregating site) resulting in a low value of this statistics, whereas a recent decline will produce an opposite result.

3. mean of within sample pairwise differences
4. mean of between sample pairwise differences
5. F_{ST} between two samples (Hudson *et al.*, 1992)

Three sample statistics :

1. Maximum likelihood coefficient of admixture (adapted from Choisy *et al.*, 2004)

2.7.3 for SNP loci

Single sample statistics :

1. proportion of loci with null gene diverty (= proportion of monomorphic loci)
2. mean gene diversity across polymorphic loci (Nei, 1987)
3. variance of gene diversity across polymorphic loci
4. mean gene diversity across all loci

Two sample statistics :

1. proportion of loci with null F_{ST} distance between the two samples (Weir and Cockerham, 1984)
2. mean across loci of non null F_{ST} distances between the two samples
3. variance across loci of non null F_{ST} distances between the two samples
4. mean across loci of F_{ST} distances between the two samples
5. proportion of loci with null Nei's distance between the two samples (Nei, 1972)
6. mean across loci of non null Nei's distances between the two samples
7. variance across loci of non null Nei's distances between the two samples
8. mean across loci of Nei's distances between the two samples

Three sample statistics :

1. proportion of loci with null admixture estimate
2. mean across loci of non null admixture estimate
3. variance across loci of non null admixture estimated
4. mean across all locus admixture estimates

2.8 Pre-evaluation of scenarios and prior distributions

This option is proposed to users since version 1.0. The purpose is to check that at least one combination of scenarios and priors can produce simulated data sets that are close enough to the observed data set. This is performed through two kinds of analyses. In the first one, a principal component analysis is performed in the space of summary statistics on at most 100,000 simulated data set and the observed data is added on each plane of the analysis in order to evaluate how the latter is surrounded by simulated data sets. In addition to this global approach, there is a second one in which each summary statistic of the observed data set is ranked against those of the simulated data set. This second analysis helps finding which aspects of the model (including prior) have been mistated. For instance, a grossly overestimated genetic distance (in simulated data sets compared to the observed one) may suggest a mispecification of the prior distribution of the time of divergence of the two involved populations or of the mean mutation rate of the markers. Using this new option before running a full ABC treatment is a convenient way to reveal mispecification of models (scenarios) and/or prior distributions of parameters (see Cornuet *et al.*, 2010, for an illustration)

2.9 Estimation of posterior distributions of parameters

Several steps are necessary to get posterior distributions of parameters. First, the normalized Euclidian distance between the observed data set and each simulated data set is computed as the sum of squared differences of summary statistics weighted by the inverse of their variance in the entire set of simulated data. For the i-th data set, the distance is :

$$d_i = \sqrt{\sum_{j=1}^{nstat} \frac{(s_{ij} - s_j^{obs})^2}{V_j}} \tag{1}$$

in which s_{ij} is the j-th summary statistics from the i-th data set, s_j^{obs} is the j-th summary statistics from the *obs*erved data set and V_j is the variance of the the j-th summary statistics across all simulated data sets. Only the closest data sets are selected for further treatments. The latter includes a weighted local linear regression step aimed at improving the posterior distributions of the parameters (Beaumont *et al.*, 2002). Basically, a multiple linear regression is performed in which summary statistics are the independent variables and parameters the dependent variables. But this regression is also *local* in the sense that more weight in the regression is given to data sets that are closest to the observed data set. This is performed by using a kernel function (the Epanechnikov kernel following Beaumont *et al.* (2002) :

$$\mathrm{K}_\delta(d) = \begin{cases} (1.5/\delta)(1-(d/\delta)^2), & t \leq \delta \\ 0, & t > \delta \end{cases} \tag{2}$$

Eventually, parameters are adjusted through this process as :

$$\phi_{ik}^* = \phi_{ik} - (\mathbf{s}_i - \mathbf{s}^{obs})\boldsymbol{\beta}_k \tag{3}$$

in which ϕ_{ik} is the k-th parameter of the i-th selected data set, ϕ_{ik}^* is the adjusted corresponding parameter, $\mathbf{s}_i$ is the row vector of summary statistics of the i-th selected data set, $\mathbf{s}^{obs}$ is the row vector of summary statistics of the observed data set and $\boldsymbol{\beta}_k$ is the transposed k-th row vector of the regression coefficient matrix.

The adjusted ϕ_{ik}^* of the selected data sets are an approximate sample of the posterior distribution of parameters (Beaumont *et al.*, 2002).

2.10 Model checking

Checking the model is crucial to statistical analysis (p161 in Gelman *et al.*, 1995). Model checking (i.e. the assessment of the ï¿œï¿œ½goodness-of-fitï¿œï¿œ½ of a model ï¿œï¿œ½ parameter posterior combination) is a facet of ABC analysis that has been so far neglected (but see Ingvarsson, 2008). Following Gelman et al. (1995; pp 159-163), we already implemented this option in $DIYABCv1.0$, to measure the discrepancy between a model ï¿œï¿œ½ parameter posterior combination and a ï¿œï¿œ½realï¿œï¿œ½ data set by considering various sets of test quantities. These test quantities can be chosen among the large set of ABC summary statistics proposed in the program. This option is based on the same kinds of analysis as section 2.7. The main difference is the set of simulated data. Whereas in section 2.7, prior distributions of parameters have been used to simulate data sets, here we use posterior distributions of the same parameters, hence simulating data from the *posterior predictive distribution.*

The first analysis is a principal component analysis in the space of summary statistics using data sets simulated with the **prior** distributions of parameters (exactly as in section 2.7) and the observed data as well as **data sets from the posterior predictive distribution** are represented on each plane of the PCA. If the model fits well the data, one should see on each PCA plane a wide cloud of data sets simulated from the prior, with the observed data set in the middle of a small cluster of datasets from the posterior predictive distribution.

In the second analysis, each summary statistics of the observed data set is ranked against the distribution of the corresponding summary statistics from the posterior predictive distribution. Summary statistics play here the role of *test statistics* (p169 in Gelman *et al.*, 1995).

Since summary statistics are generally not sufficient, it is advised to use different sets of summary statistics to compute the posterior distribution of parameters on one hand and to check the model on the other hand (see Cornuet *et al.*, 2010). This has been implemented in DIYABC.

2.11 Measures of performances

As stressed in previous studies (e.g. Excoffier *et al.*, 2005), the ABC appproach provides an efficient way of assessing its own performances for estimating posterior distributions of parameters. The reference table, the building of which represents generally 95 to 99% of the computing time, can be reused to analyse pseudo-observed (test) data sets obtained through simulation with known values of parameters. It is then rather quick and easy to evaluate the performance of the method for parameter estimation by computing statistics such as estimation biases or mean square errors.
These measures of performance have been fully integrated into DIYABC. The performance measures computed by DIYABC are :

the average relative bias : the difference between the point estimate (e) and the true value (v)divided by the true value, $\frac{1}{n}\sum_{i=1}^{n}\frac{e_i-v_i}{v_i}$, averaged over the n test data sets,

the square Root of the Relative Mean Square Error (RRMSE) : the square root of the average square difference between the point estimate and the true value, divided by the true value, $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\frac{e_i-v_i}{v_i})^2}$

the square Root of the Relative Mean Integrated Square Error (RRMISE) : the square root of the average (over test data sets) of the integrated square error (measured on each test data set) divided by the true value, $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\frac{\sum_{j=1}^{m_i}(x_{ij}-v_i)^2}{m_i v_i^2})}$, x_{ij} and m_i being the sampled values and the sample size of the posterior distribution in the i-th test data set, respectively.

the Relative Mean Absolute Deviation (RMAD) : the average (over test data sets) of the mean absolute deviation (measured on each data set), divided by the true value, $\frac{1}{n}\sum_{i=1}^{n}(\frac{\sum_{j=1}^{m_i}|x_{ij}-v_i|}{m_i|v_i|}$

the factor 2 :the proportion of test data sets for which the point estimate is at least half and at most twice the true value.

the Relative Median Bias (RMB) : the 50% quantile of the bias (measured on each test data set) divided by the true value. The bias is computed respectively for each point estimate

the Relative Median Absolute Deviation (RMedAD) : the 50% quantile (over test data sets) of the median (over each data set) of the absolute difference between each value of the posterior distribution sample and the true value divided by the true value.

the Relative Median of the Absolute Error (RMAE) : the 50% quantile (over test data sets) of the absolute value of the difference between the point estimate (in each data set) and the true value divided by the true value.

DIYABC considers the following three point estimates : mean, median and mode of the ϕ_{ik}^* (sample of the posterior distribution of each parameter), as defined in subsection 1.7.
Concerning the true value (v) appearing in the above formulae, DIYABC offers three possibilities :

1. All values v are fixed by the user. If any one of these values is outside the limits given to the prior for the corresponding parameter, a warning message is issued but the analysis can proceed if needed.

2. All values v are drawn from prior distributions. These distributions can also be different from those of priors. They may even not be overlapping (no warning message is issued whatever the user's choice).

3. All values v are drawn from posterior distributions (in order to obtain accuracy measures conditionally to the observed dataset).

If you want to fix some parameter values and draw the other from distributions, choose the second option and give the same desired values as minimum and maximum for those fixed parameter values.

In order to better assess the information brought by genetic data, DIYABC provides a double estimate of all these bias/precision statistics. As expected, the first one is based on genetic data given in the data file. The second one is computed as if there was no genetic information, *i.e.* estimates are based only on parameter priors. Technically, a sample of parameter values is drawn at random from the reference table. This sample of the same size of the sample of posterior values is used in place of the latter in all computations.

2.12 Comparison of scenarios

The ABC approach can also be used to compare possible scenarios for the same data file through the computation of the posterior probabilities of each scenario and this option is naturally implemented in DIYABC.

2.12.1 Reference table

First, the reference table can include as many scenarios as desired. By default, the prior probability of each scenario is uniform, that is each scenario will have approximately the same number of simulated data sets. But, if for any reason, one wants a different prior probability for each scenario, there is the possibility to do so.

Scenarios are drawn according to their own prior probability and then only parameters that are defined for the drawn scenario are generated from their respective prior distribution. Scenarios may or may not share parameters.
When conditions apply to some parameters (see subsection 2.4), the program provides the possibility of choosing between two options :

1. parameter sets are drawn in their respective prior distributions until all conditions are fulfilled.
2. a single parameter set is drawn and only if all condition are fulfilled, the simulation is performed and the data set is recorded in the reference table.

When there is only one scenario, both options are equivalent, although in option 2, there might be less simulated data sets that are recorded than one asked. When there is more than one scenario, the second option can be viewed as a way to set prior probabilities on scenario that result from imposed conditions on parameters (see Miller *et al.* (2005) for an example).

2.12.2 Posterior probability of scenarios

The program DIYABC provides two estimates of the posterior probability of each scenario :

a emphdirect estimate : This is simply the number of times that a given scenario is found in the first n_δ simulated data sets once the latter, produced under several scenarios, have been sorted by ascending distances to the observed data set (*i.e.* the "closest" simulated data sets).

a logistic regression estimate : Following M.A. Beaumont's suggestion (Fagundes *et al.*, 2007; Beaumont, 2008), a polychotomic weighted logistic regression is performed on the first n_δ data sets with the proportion of the scenario as the dependent variable and the differences between observed and simulated data set summary statistics as the independent variables. The intercept of the regression (corresponding to an identity between simulated and observed summary statistics) is taken as the point estimate. In addition, 95% confidence intervals are computed (Cornuet *et al.*, 2008).

Since both estimates are dependent upon the chosen threshold (δ), the program provides a range of 100 estimates for the direct approach (for each one 100-th of n_δ between 0 and n_δ) and up to 10 estimates for the logistic regression estimates (e.g. one estimate for $kn_\delta/10$ with $k \in [1, 2, ...10]$ when the number of analyses is set to 10). These estimates are represented in two graphs, one for each kind of estimate. These two graphs can be printed and/or saved (in *svg*, *jpg*, *png* or *pdf* format). Values can also be output as a text file. In $DIYABCv2.0$, a new possibility is offered to the user that may be useful when dealing with many summary statistics and many scenarios. In this particular case, the logistic regression has to deal with large matrices and the amount of needed memory on one hand and the computation time on the

other hand can become problematically large. An approximate solution is to replace summary statistics by the components of a linear discriminant analysis which reduces the number of independent variables to the smallest of number of summary statistics and scenarios. Although the result is only approximate, it can be a useful guide in some specific cases. The gain in time can be large. For instance, the time can be reduced by a 100X factor (Estoup *et al.*, 2012).

2.12.3 Confidence in scenario choice

The program DIYABC offers a last option that allows one to evaluate the confidence in a scenario choice. To do so, we simulate test datasets (or pods), apply the same procedure for estimating their respective posterior probabilities and measure the proportion of times the right scenario has the highest posterior probability. More specifically DIYABC proposes three main options :

(i) Compute confidence in scenario choice drawing scenario-parameter combinations into posterior distributions (cf. Posterior based error). Computing error rate conditionally to the observed dataset (i.e. focusing around the observed dataset by using the posterior distributions) provide a more relevant estimation of our ability to choose the true scenario in the vicinity of the observed dataset (which is the location of prime interest in the vast data space defined by the prior distributions) than blindly computing accuracy indicator over the whole prior space.

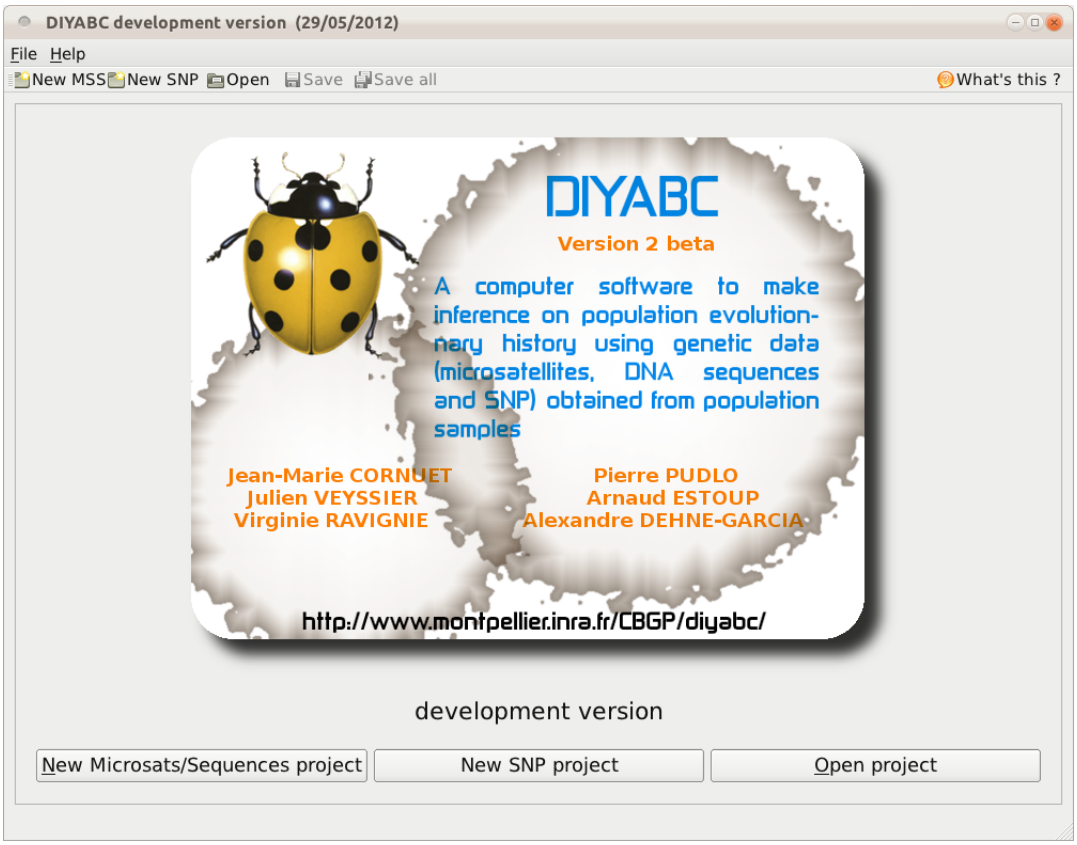
(ii) Compute confidence in scenario choice drawing scenario-parameter combinations into prior distributions (cf. Prior based error). Prior based error computation provides an estimate of a global error level over the whole (and usually huge) prior data space. Such computation can be useful for comparisons with the above posterior error rate, to focus investigation on a particular scenario and to select the best classifier and/or set of summary statistics (Pudlo et al. 2015). Two sub-options are proposed for the computation of prior based errors:

- Global (prior error rate) in which pods are drawn from a random sample of scenario ID and parameter values in the prior distributions;

- Scenario specific (prior error rate) in which pods are drawn from parameter prior distributions under a GIVEN scenario. This corresponds to the confidence in scenario choice option that was initially available in the previous version of the program (DIYABC v2.0). In this sub-option parameter values can also be fixed to given values.

3. The Graphic User Interface

When launching the GUI, the home screen appears like this :



You can already notice that DIYABC works with projects. This notion is new to version 2 of DIYABC. It is explained in subsection 3.1.

3.1 What is a DIYABC Project ?

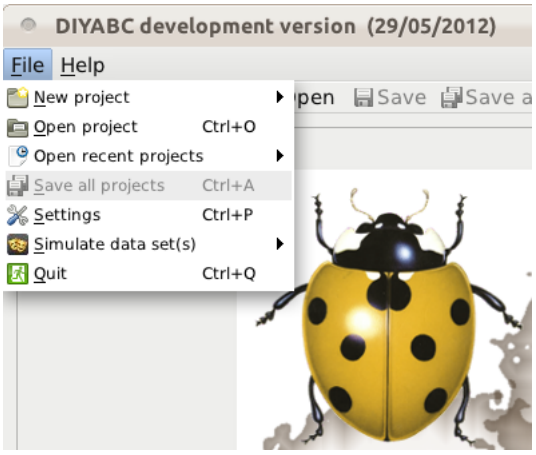
A *DIYABC* project is a unit of work materialized by a specific and unique directory. A project is defined by at least one observed data set and one reference table header file. These files are located in the *Project directory* which name includes an identifier, the date of creation and a number (between 1 and 100).

The header file, always named `header.txt`, contains all information necessary to compute a reference table associated with the data : i.e. the scenarios, the scenario parameter priors, the characteristics of loci, the loci parameter priors and the summary statistics to compute. As soon as the first records of the reference table have been saved in the reference table file, always named `reftable.bin` and also included in the project directory, the project is "locked". This means that the header file can not be changed anymore. If one needs to change a scenario or a parameter prior, or a summary statistics, a new project needs to be defined. This is to guarantee that all subsequent actions performed on the project are in coherence with the current data and header files. It is of course strongly advised NOT to move files among projects. Incidentally, the `header.txt` file is only built when the project has been saved, the information progressively input by the user being saved in a series of temporary files.

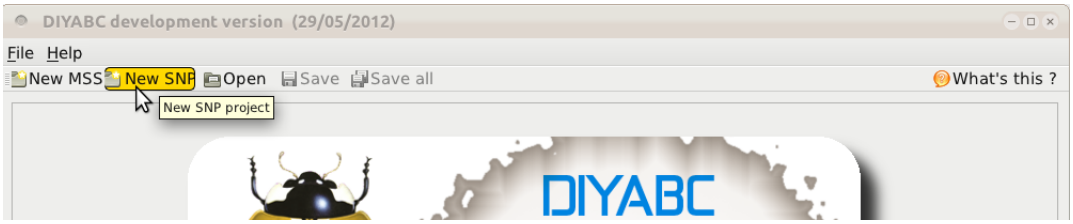
Once a sufficiently large reference table has been simulated, analyses can be performed. Their different output files are copied to the *analysis* directory included in the project directory, and containing as many directories as analyses performed. Hence, it is now much easier to know with certainty the conditions of each analysis.

3.2 Options of the home screen

The home screen above has two menus and several buttons.
Let's start with the menus. Below are shown all submenus :



The `File` menu has seven options, namely `New project`, `Open project`, `Open recent projects`, `Save all projects`, `Settings`, `Simulate data set(s)` and `Quit`. All are self explanatory.
The `Help` menu has two options : `About DIYABC` which opens up a small window providing the names and address of the authors and `Show logfile` which gives access to a logfile viewer in which are recorded all actions and messages about the execution of the GUI.
Just below the menu are five shortcuts to main `File` menu options.



On the right, the field `What's this ?` is an another way to get help on a specific GUI object :

Eventually, below the logo, there are three buttons which are duplicate shortcuts :

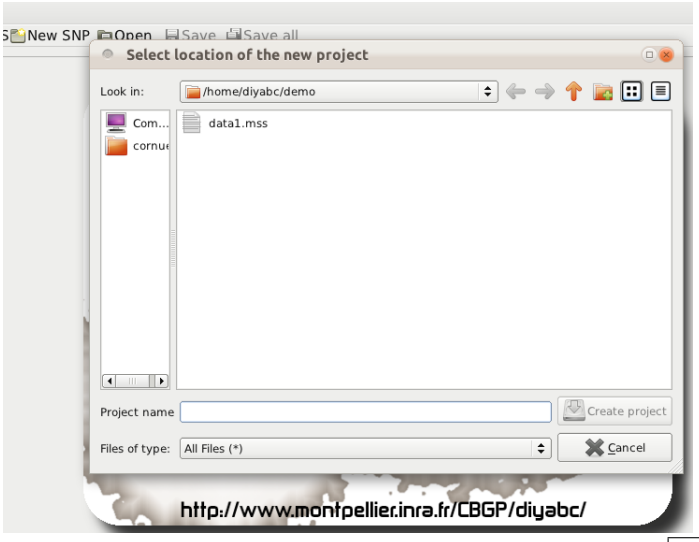
3.3 Defining a new project

Defining a new project requires different steps which are not the same whether the data are SNPs or microsatellites/DNA sequences (MSS). Let start with an MSS project : click on one of the following :

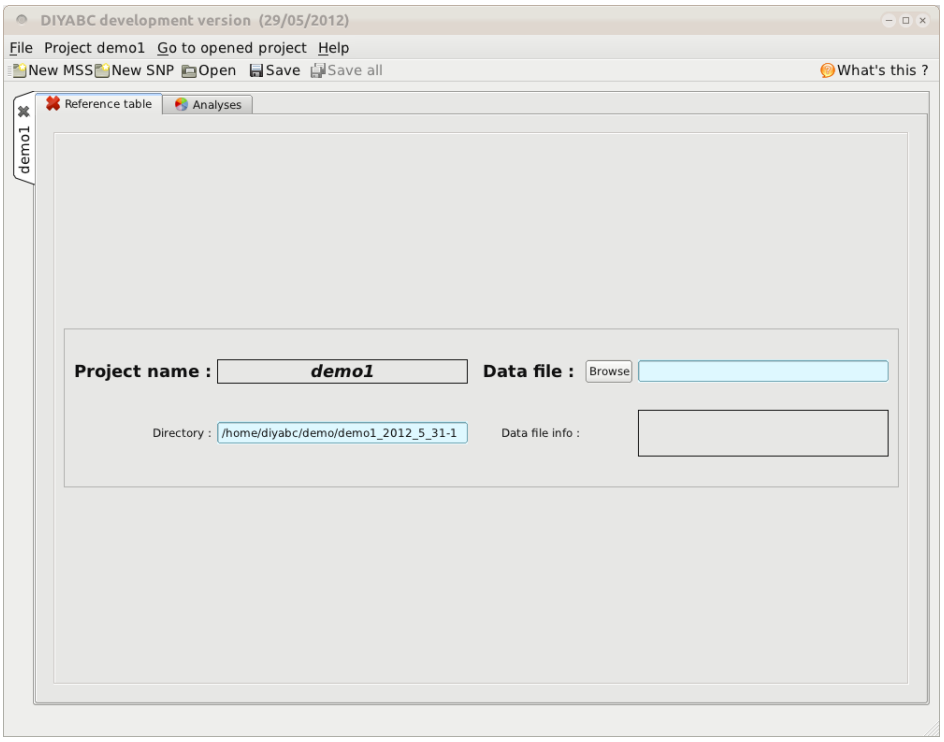
- `File` menu > `New project` > `Microsatellites and/or sequences`
- the menu shortcut `New MSS`

- the bottom left button New Microsat/Sequence project

or press simultaneously the `Control` and `M` keys.
A new window appears in which the user can choose a location and a name for the new project as shown below :



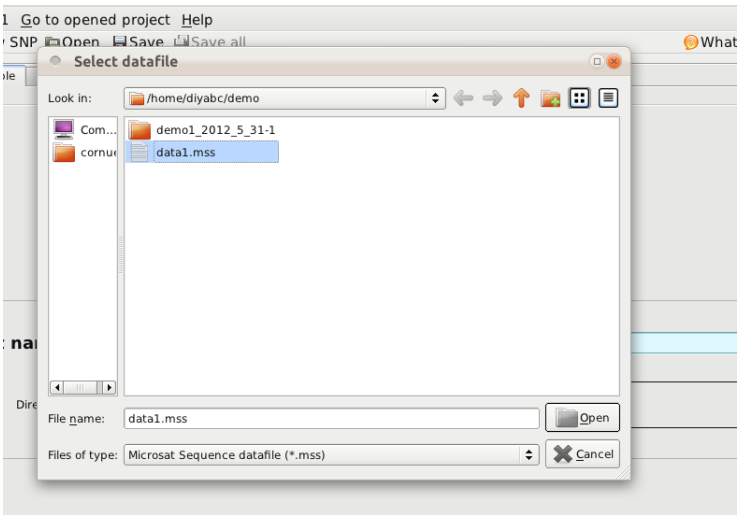
Let's enter **demo1** as the project name and click on the Create project button.
The following screen appears :



The **demo1** project and all its future files will be located in the directory `demo1_2012_5_31-1`.

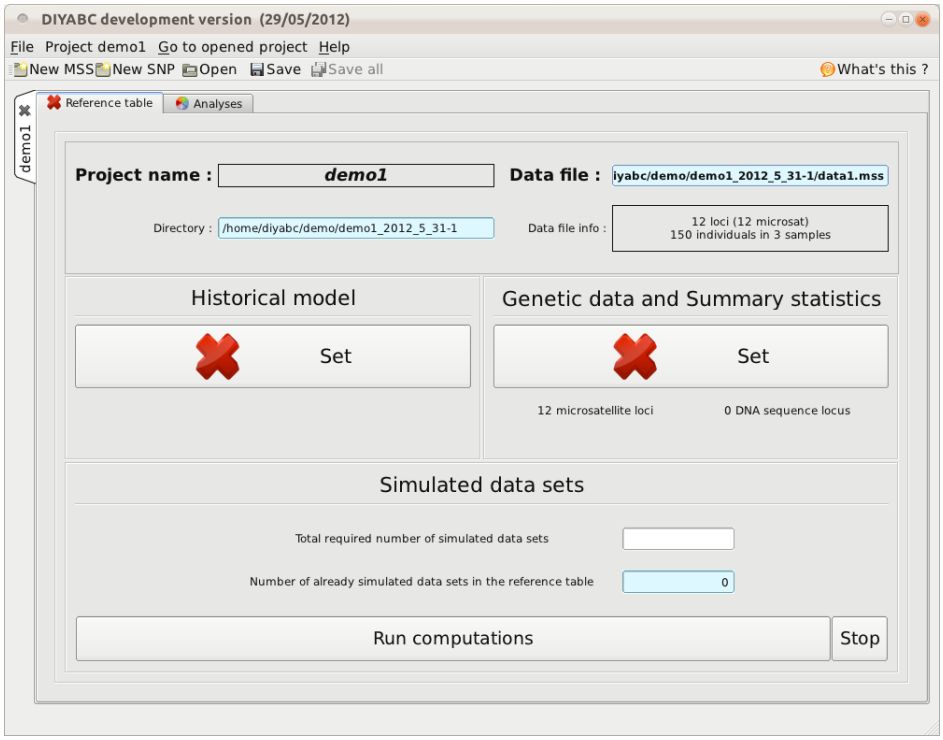
3.3.1 Step 1 : choosing the data file

We next need to choose the data file of the project. This is performed by clicking on the corresponding Browse button (previous screen). The usual file browsing screen appears (below) and one has to select a Genepop format data file, here `data1.mss`.



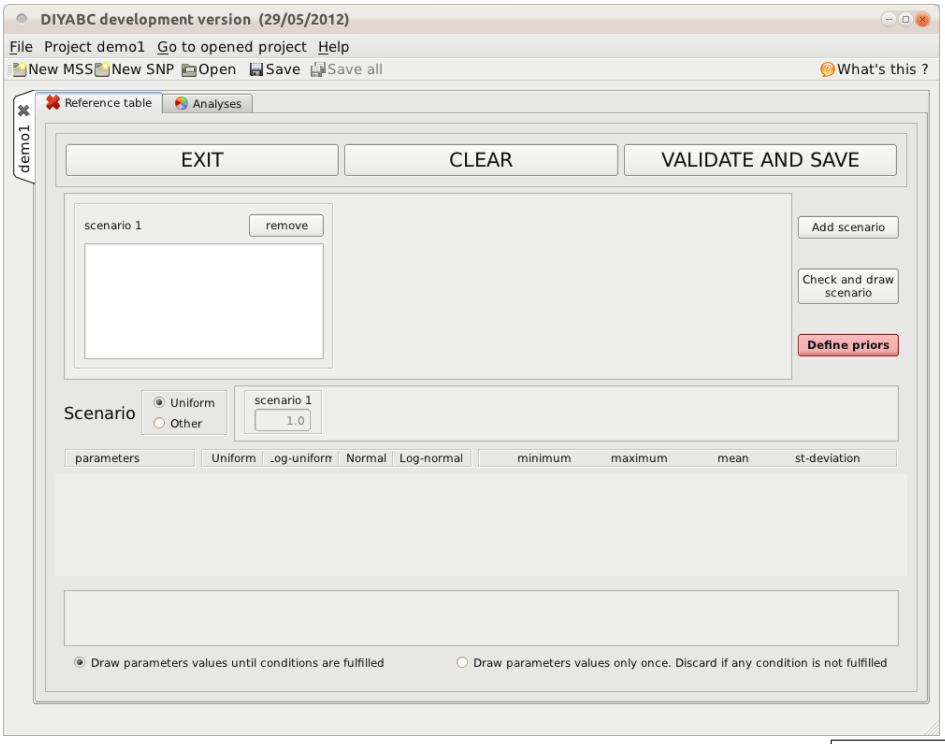
Clicking on the Open button leads to the following screen with the edit field filled with the name of the data file and some characteristics of this data file appearing on the screen (number of loci, individuals and samples).

Below these fields are two panels indicating that we need to provide information about the Historical model (left panel) and about the Genetic data and associated Summary statistics (right panel). The red crosses on both panels will change to green checks once the corresponding information will be completed.

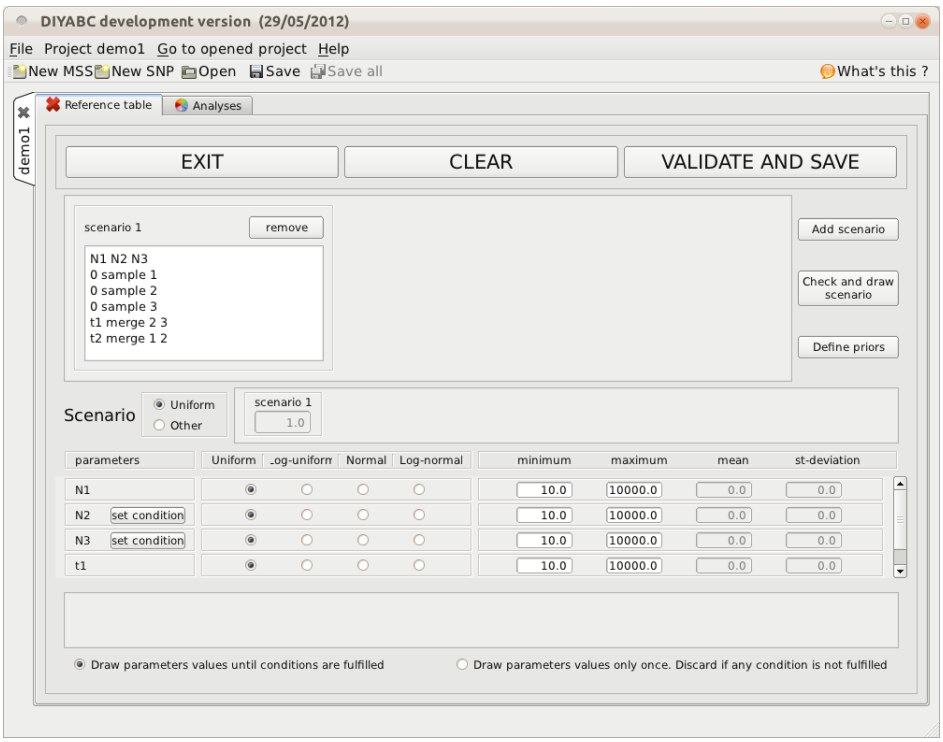


3.3.2 Inform the Historical model

Click on the corresponding Set button. The following screen, familiar to users of previous versions, appears:



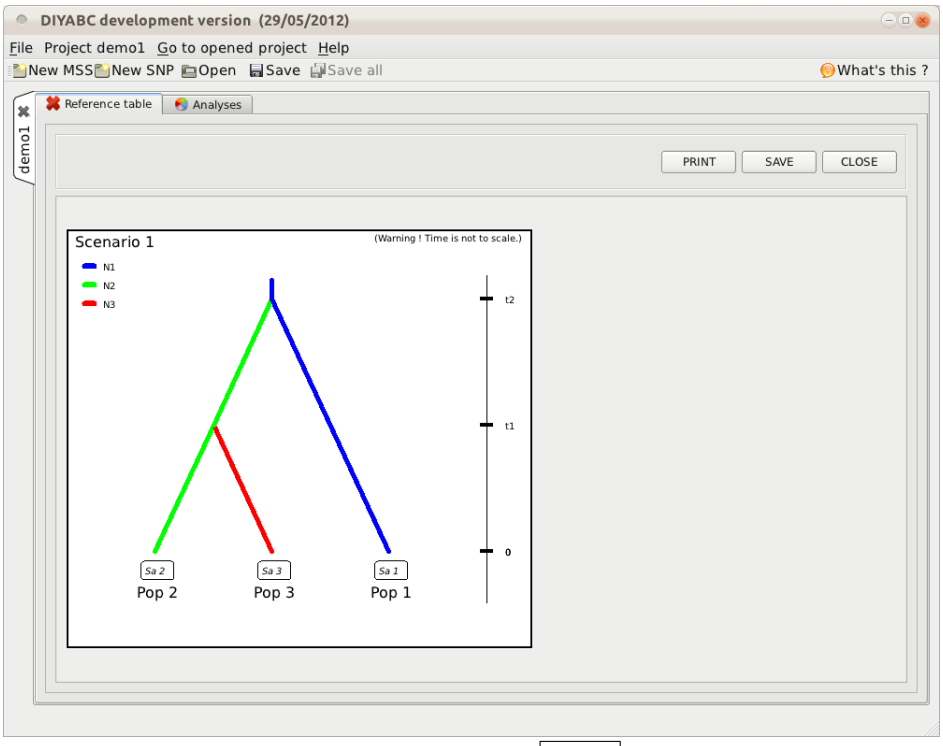
Let's enter a simple scenario in scenario 1 edit window and click on the Define priors button. We get this :



The parameter prior frame allows to choose the prior density of each parameter. A parameter is anything in the scenario that is not a keyword (here `sample` and `merge`), nor a numeric value. In our example scenario, parameters are hence : `N1, N2, N3, t1` and `t2`. In our example, we need to set the priors on `t1` and `t2` such that `t2`> `t1`. We can do it either by using the `set condition` button or by playing with the minimum and maximum values of the two parameters. It is worth stressing that the

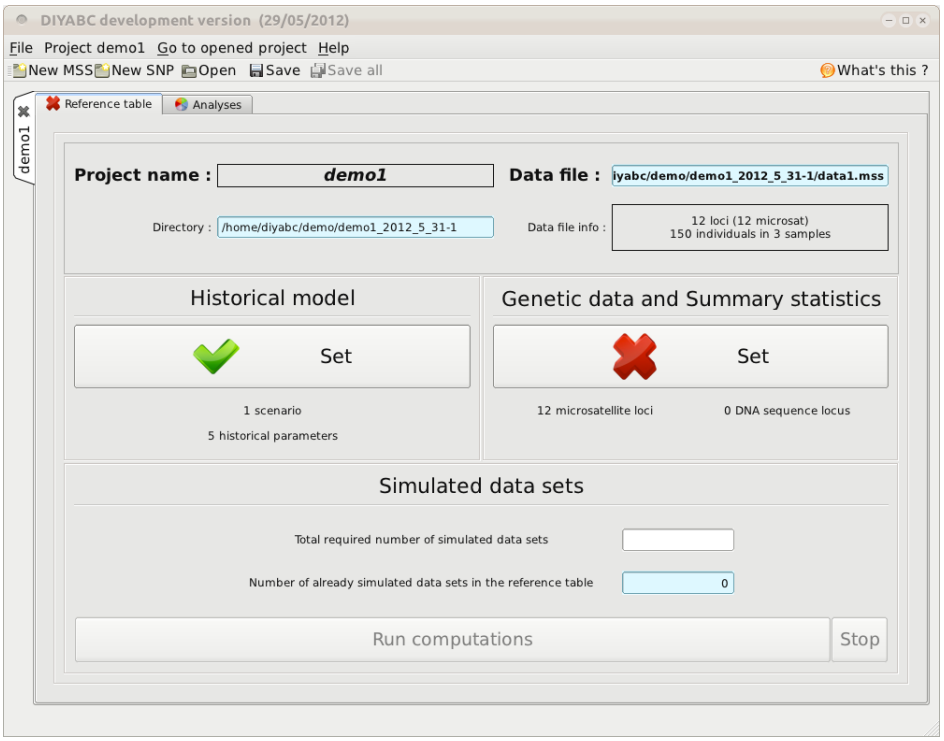
omission of such conditional constraints on merge times (cf. a population needs to exists in the past to allow coalescence events in it) is one the most frequent implementation error made by DIYABC users. If forgotten a Â gene genealogy failure Â message pointing to the problematic scenario will appear when launching simulations. Note that the occurrence of a too large number of time conditional constraints within a scenario may substantially slow down simulations as a valid t parameter vector will be retain and run only once all conditions are fullfiled.

If we click on the Check scenario button, the logic of the scenario is checked and if it is found OK, and if the scenario is drawable, the drawing appears on a new frame :



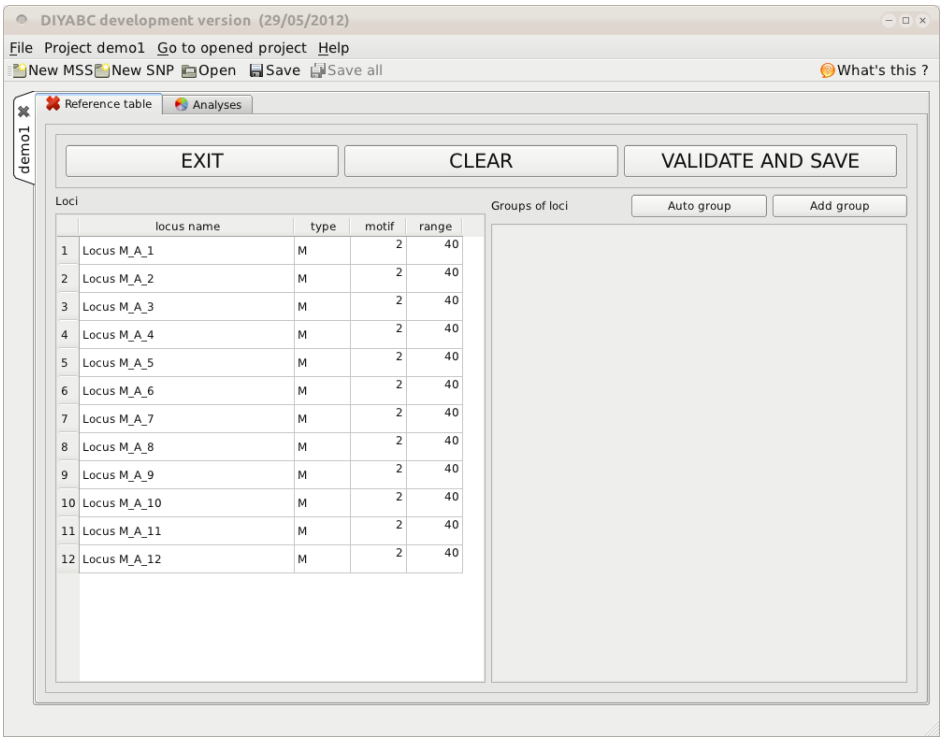
The scenario can be saved by clicking on the SAVE button. The frame can be close by clicking on the CLOSE button.

Since the scenario has been checked, we can validate and save the historical model by clicking on the VALIDATE AND SAVE button (bottom screen of p 21). We then go back to the project screen in which the historical model has now received the green check sign.


DIYABC development version (29/05/2012)
File Project demo1 Go to opened project Help
New MSS New SNP Open Save Save all
What's this ?
demo1
Reference table
Analyses
Project name : demo1
Data file : iyabc/demo/demo1_2012_5_31-1/data1.mss
Directory : /home/diyabc/demo/demo1_2012_5_31-1
Data file info :
12 loci (12 microsat)
150 individuals in 3 samples
Historical model
Set
1 scenario
5 historical parameters
Genetic data and Summary statistics
Set
12 microsatellite loci
0 DNA sequence locus
Simulated data sets
Total required number of simulated data sets
Number of already simulated data sets in the reference table
0
Run computations
Stop

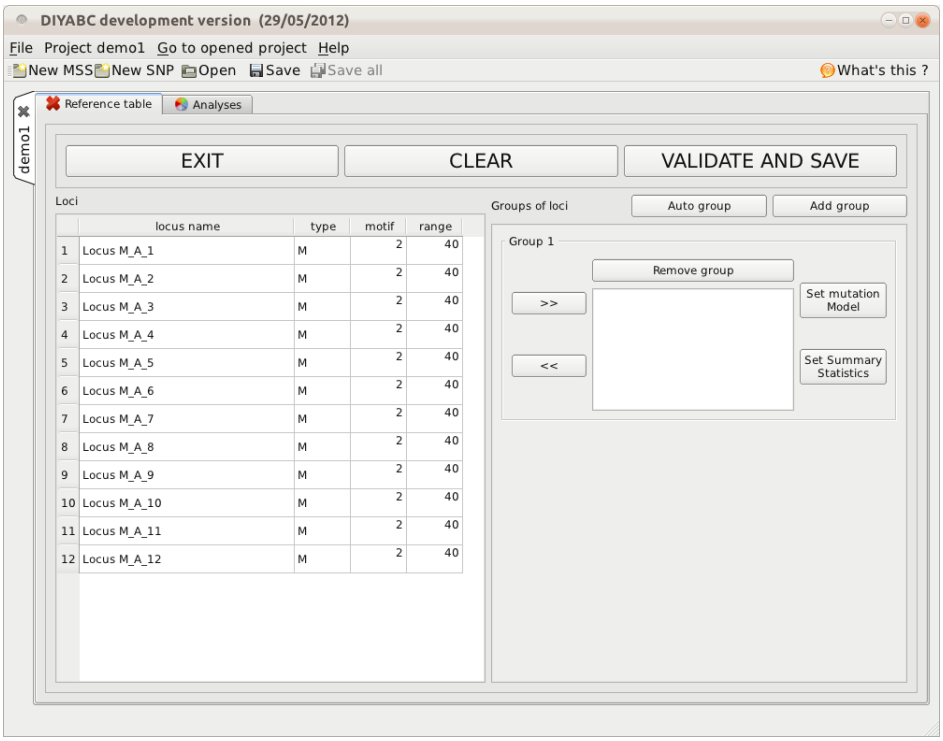
3.3.3 Inform the Genetic model

Click on the corresponding Set button. We get the following screen :

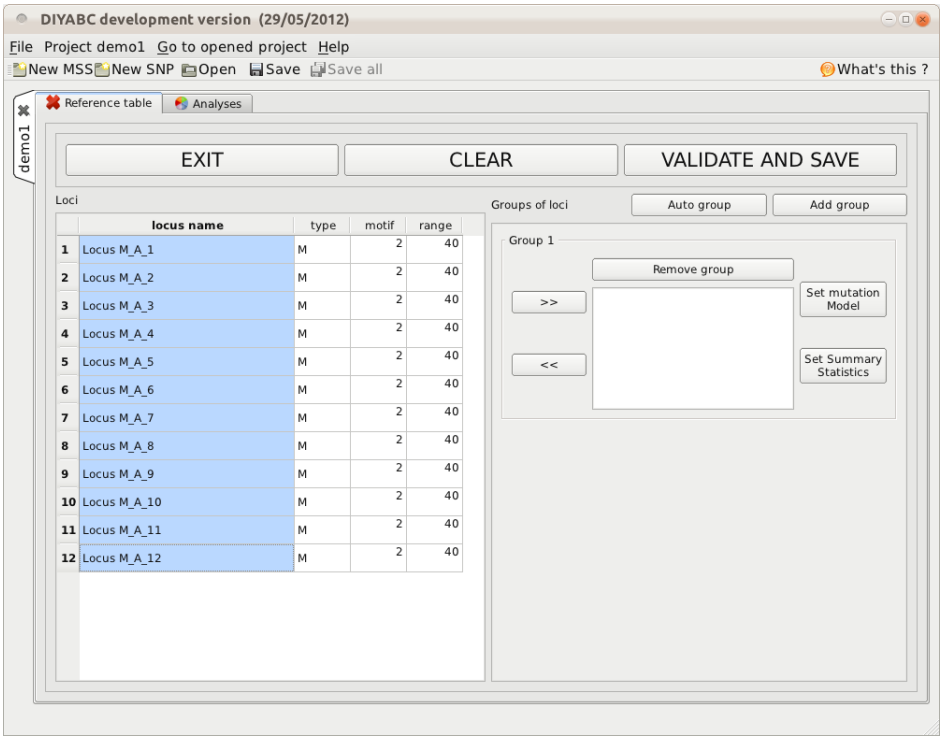


On the left part of the screen, there is the list of loci, with their type (M for microsatellites or S for DNA sequences) and the motif size and allelic range for microsatellite loci only. Actually, the values for motif size and allelic range are just default values and do not necessarily correspond to the actual data. The user who knows the real values for its data is required to set the correct values at this stage. If the range is too short to include all values observed in the analysed dataset, a message appears in a box asking to enlarge the corresponding allelic range. Note that the allelic range is measured in number of motifs, so that a range of 40 for a motif length of 2 bp means that the difference between the smallest and the longest alleles should not exceed 80 bp. It is worth stressing that the indicated allelic range (expressed in number of continuous allelic states) corresponds to a potential range which is usually larger than the range observed from the analyzed dataset (cf. all possible allelic states have usually not been sampled). In practice it is difficult to assess the actual microsatellite constraints on the allelic range; to do that one needs allelic data from several distantly related populations/sub-species as well as related species which is rarely the case (see Pollock *et al.*, 1998); (Estoup *et al.*, 2002). We achieved a meta-analysis from numerous primer notes documenting the microsatellite allelic ranges of many (i.e. >100) different species (and related species). We used the corrective statistical treatment on such data proposed by (Pollock *et al.*, 1998). Our results pointed to a mean microsatellite allelic range of 40 continuous states (hence the default allelic range value of 40 mentioned in the program). We also found, however, that range values greatly varied among species and among loci within species (unpublished results). We therefore recommend to use the following pragmatic behaviour when considering the allelic range of your analysed microsatellite dataset: (i) if the difference in number of motif of your locus is <40 motifs in the analysed dataset then leave the default allelic range value of 40. (ii) if the difference in number of motif of your locus is >40 motifs in your dataset then take Max_allele_size − Min_allele_size)/motif size + say 10 additional motifs to re-define the allelic range of the locus in the corresponding DIYABC panel (e.g. (200 nu − 100 nu)/2 + 10 = 50 + 10 = 60 as allelic range).

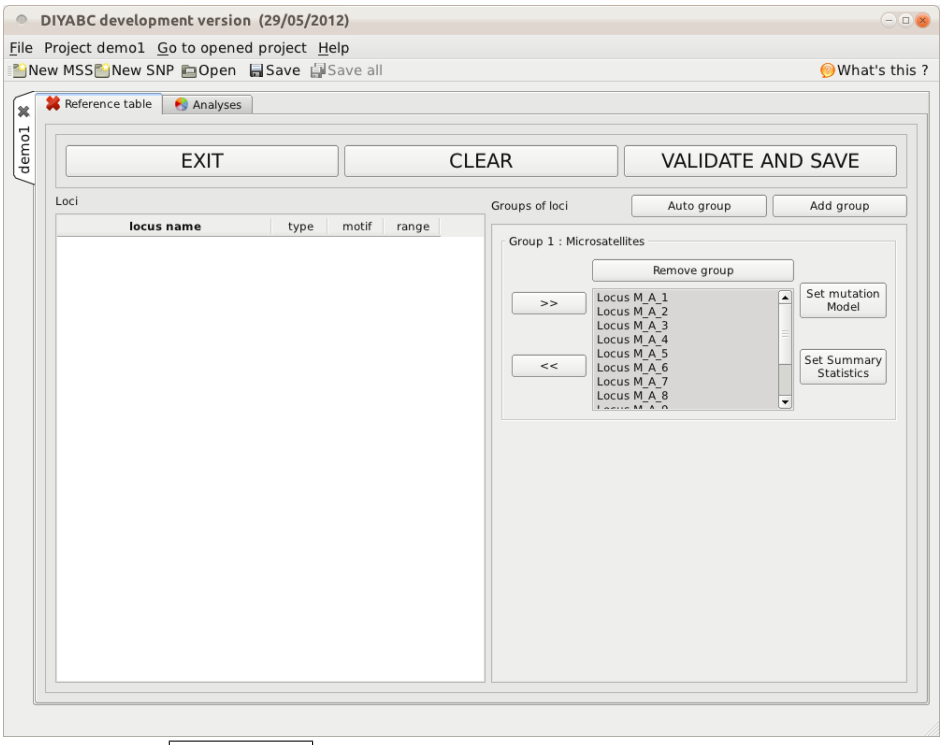
We then need to define at least one group of loci by clicking on the Add group button. We get this :



Suppose we want all loci in the same group because we consider that they all have similar mutational modalities. We select them like in any table, extending the selection with the `Shift` and `Control` keys (see below) :

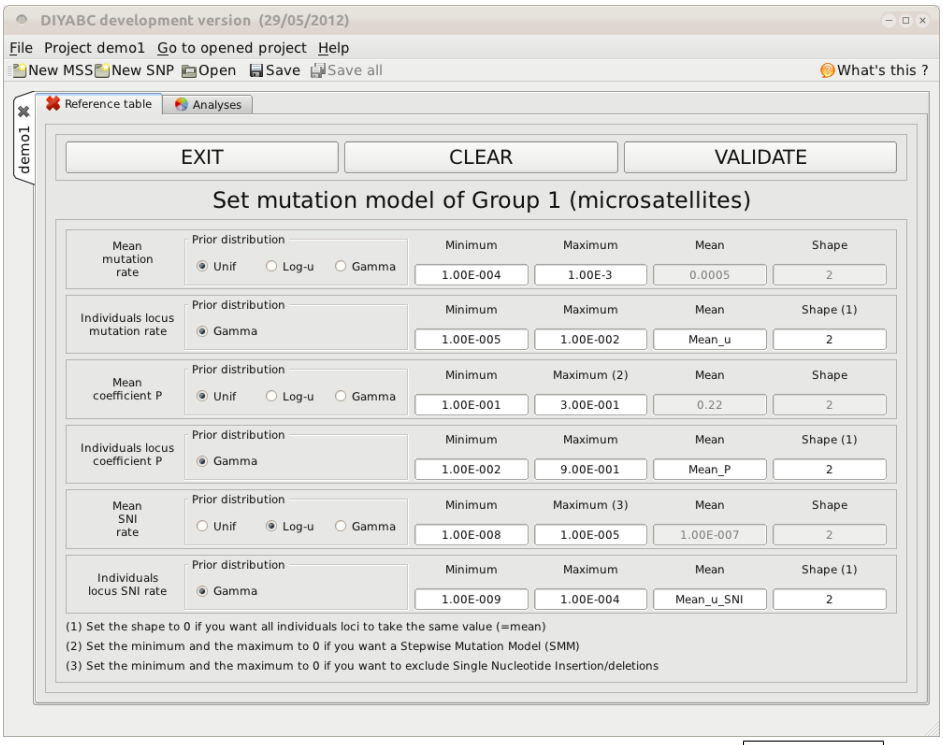


and then pressing the $\boxed{>>}$ button :

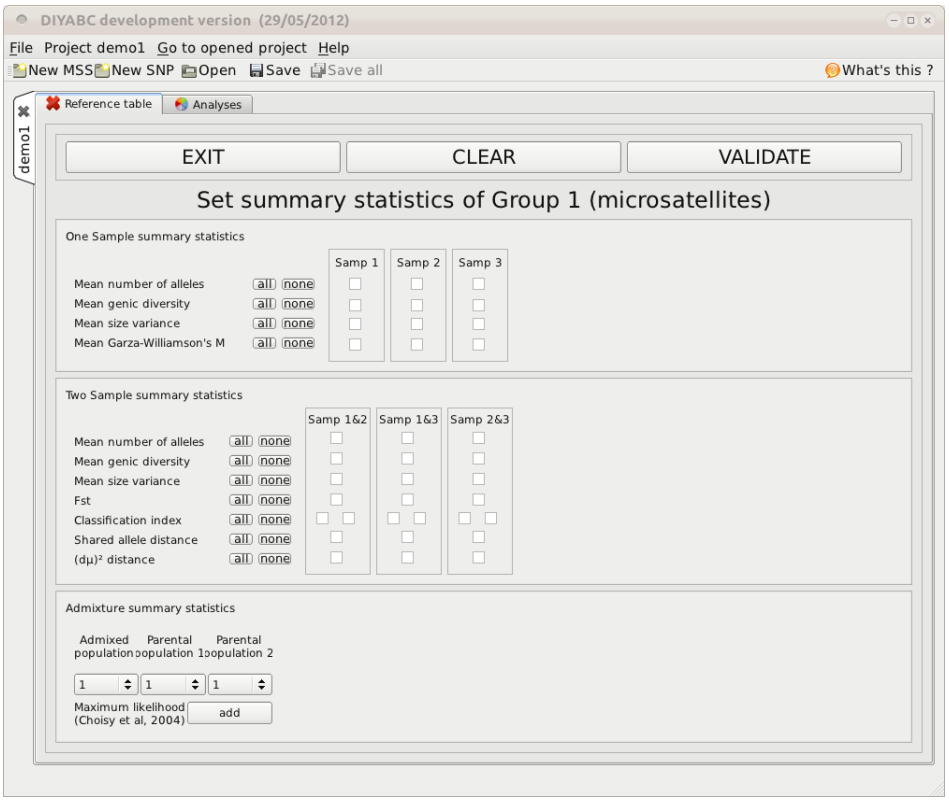


Note that the Auto group button would have produced the same result of putting all the microsatellite loci in the same group.

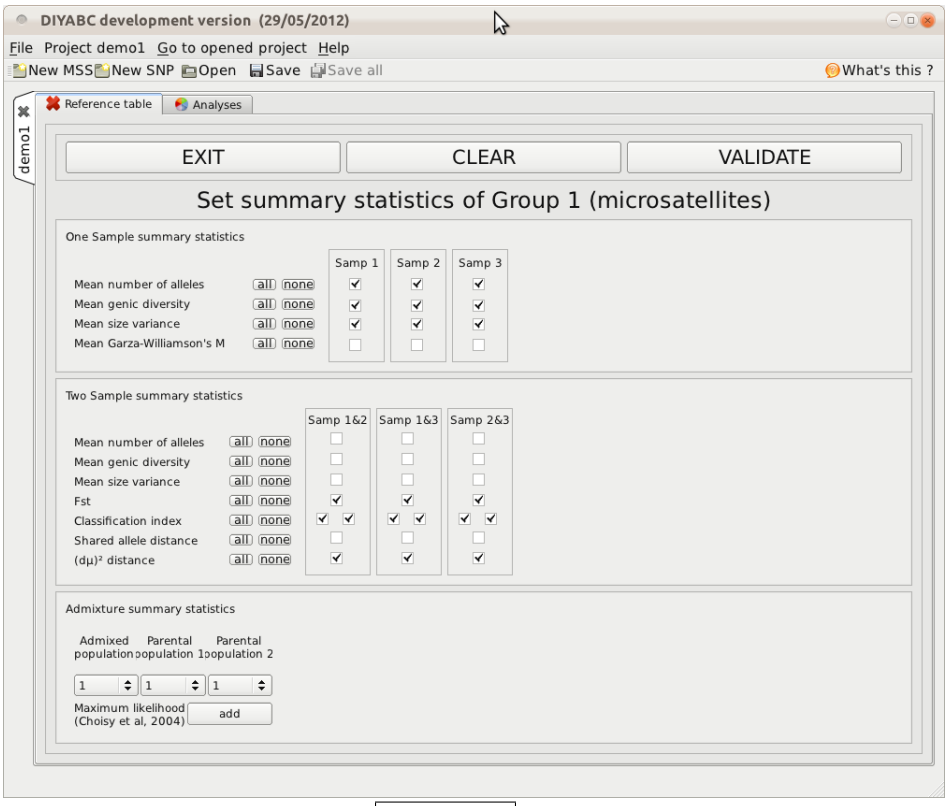
We then need to define the mutation model and the summary statistics of the locus group. Clicking on the Set mutation model button, the following screen appears :



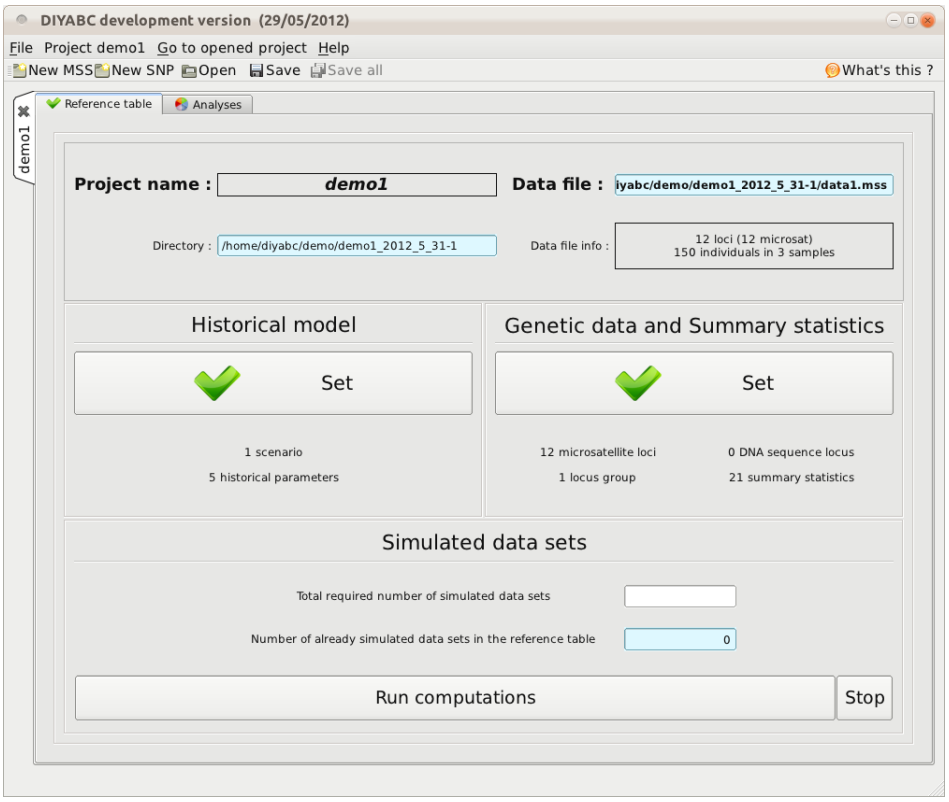
Once the mutation model of Group 1 is defined, we click on the VALIDATE button to go back to the previous screen. Clicking on the Set Summary statistics button, we get the following screen :



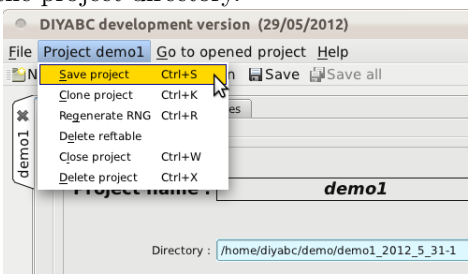
We define summary statistics by checking the corresponding boxes :



Once finished, we click on the VALIDATE button to go back to the screen of p24. Now, we can validate also this screen which brings us back to the screen of p22. The latter looks now like this :

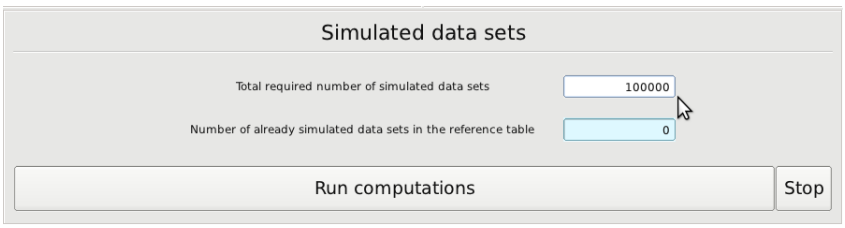


At that moment, the project directory includes the following files : a copy of the data file, and four configuration files : `conf.analysis`, `conf.gen.tmp`, `conf.hist.tmp`, `conf.tmp`. Note that the project is not yet saved. To save the project, we need either to save it explicitly by using the `File` menu (see below) or to start simulating data sets (next section). Saving the project results in saving the `header.txt` file in the project directory.

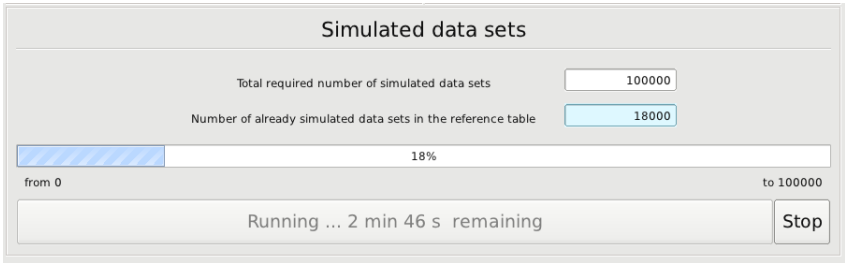


3.4 Building the reference table

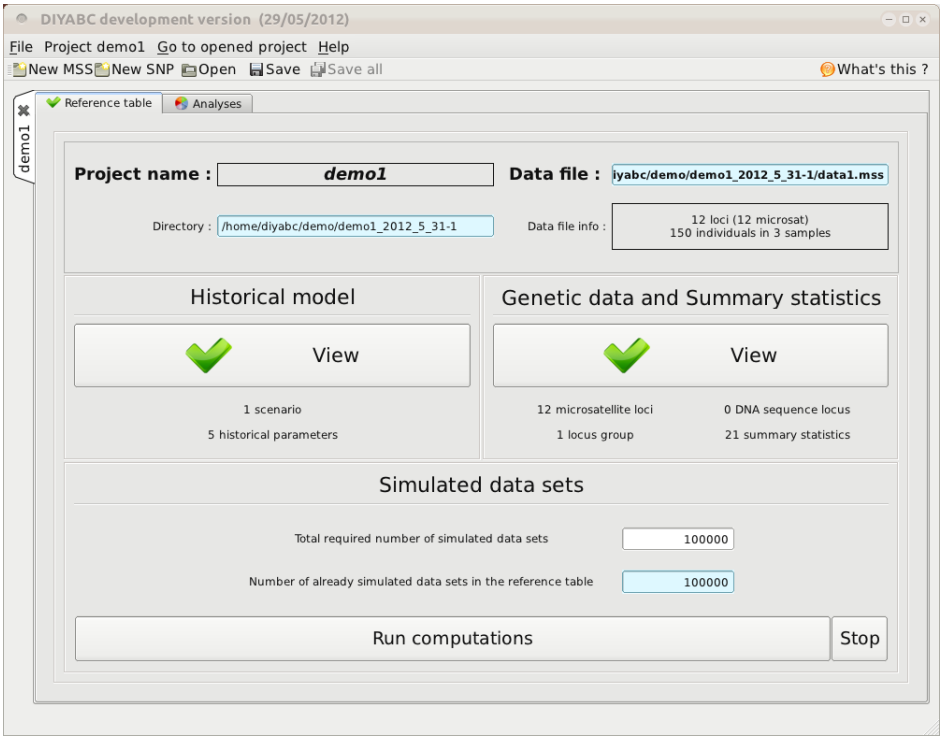
Keeping on the current screen, indicate the required number of data sets to simulate for the reference table :



Then click on the Run computations button. If things go well, you will soon see the progress both into the edit window "Number of simulated data sets in the reference table" and in the progress bar below. Also, you have an estimate of the remaining time (at the left of the Run computations button):

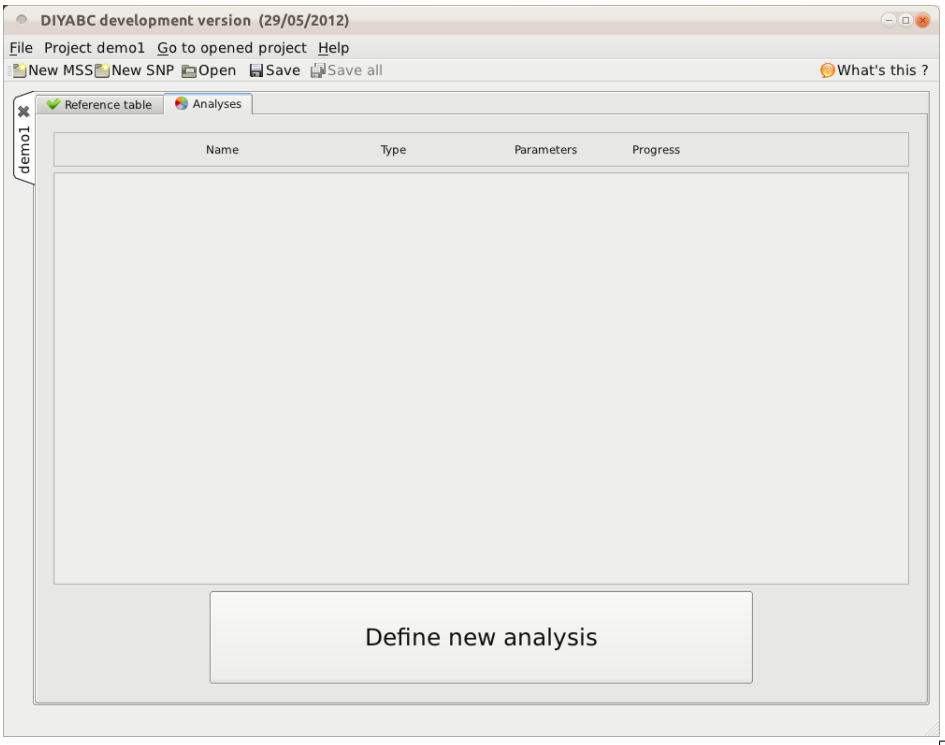


When the computation is finished, the screen looks like this :

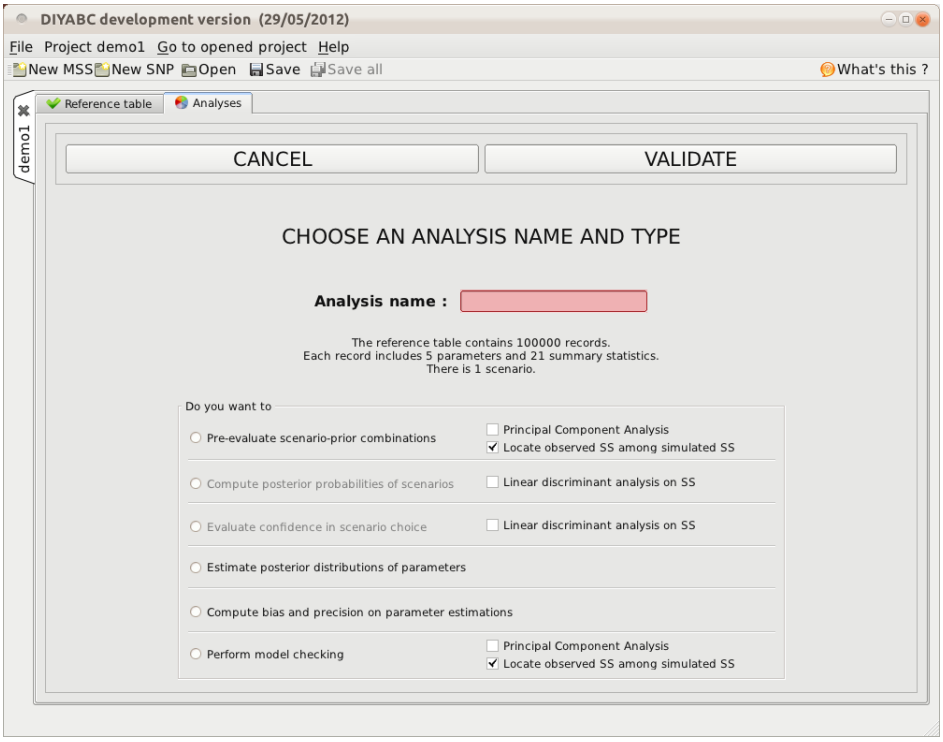


3.5 Performing analyses

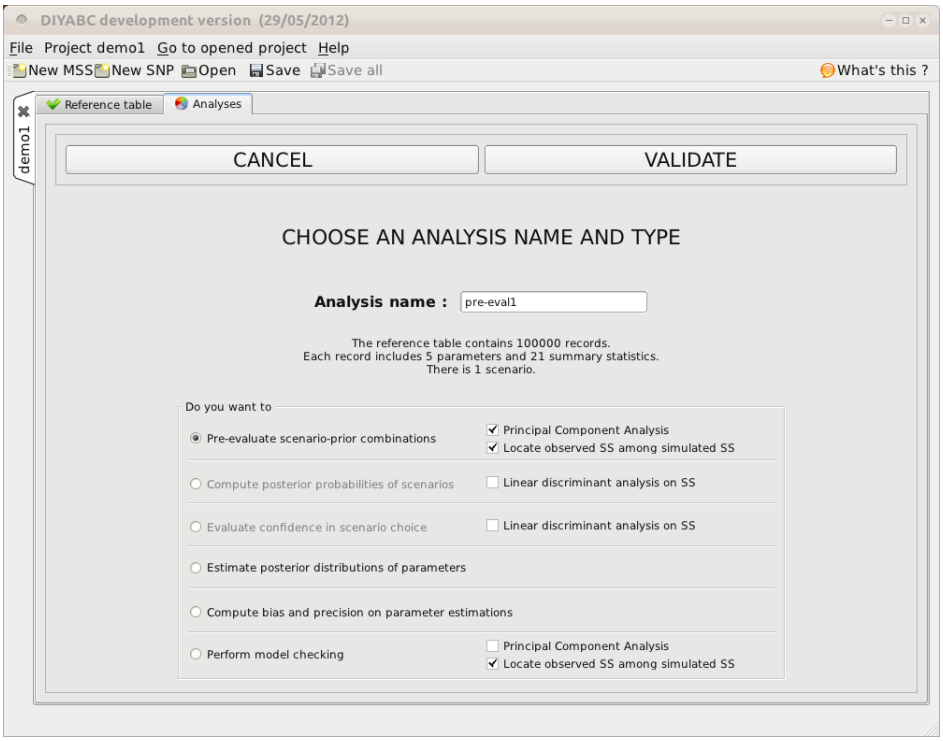
We have now eveything necessary to perform analyses. The current screen shows two tabs : `Reference table` and `Analyses`. Let's click on the `Analyses` tab. We get this new screen :



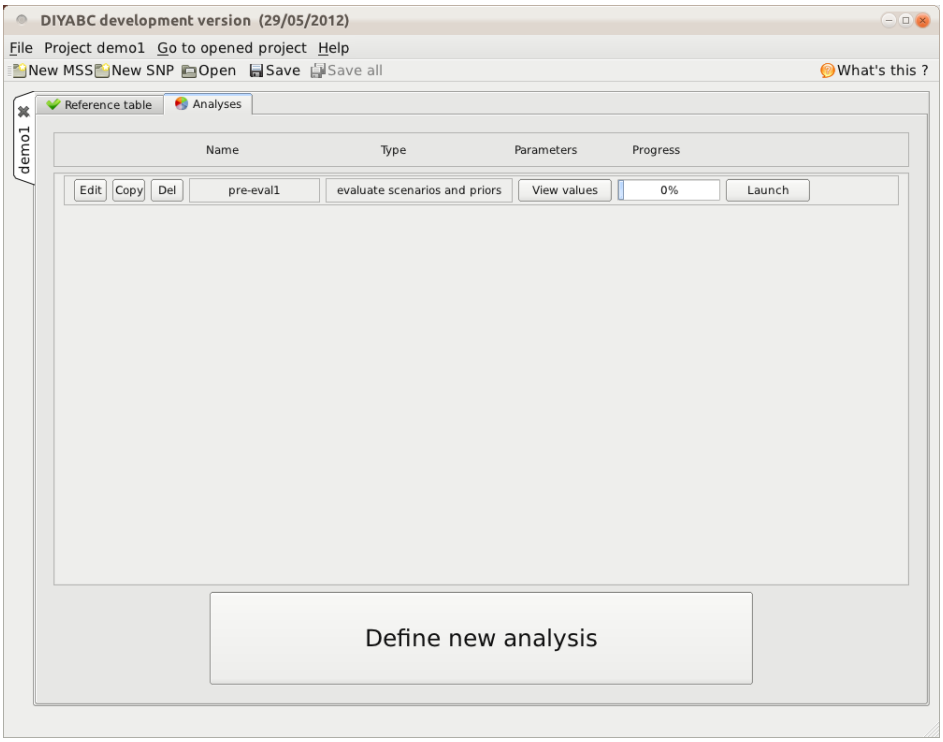
First, we need to define the analysis we want to perform. So we click on the Define new analysis button and get this new screen :



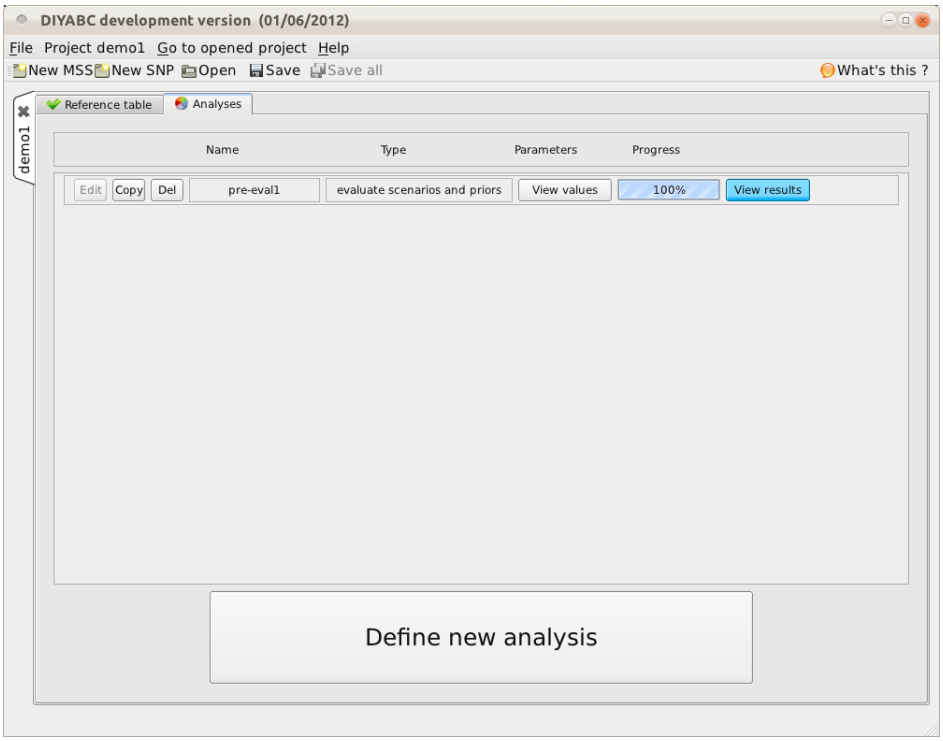
We need to choose among the six possible types of analyses (actually, only four of them are possible, since the reference table includes a single scenario). We decide to first check whether the model (scenario and parameter prior definition) is off the target or not. This can be appreciated through the analysis denominated `Pre-evaluate scenario prior combination`. To illustrate the result, we also ask for a principal component analysis by checking the corresponding square. Eventually, we give the name of **pre-eval1** to this first analysis. The screen now looks like this :



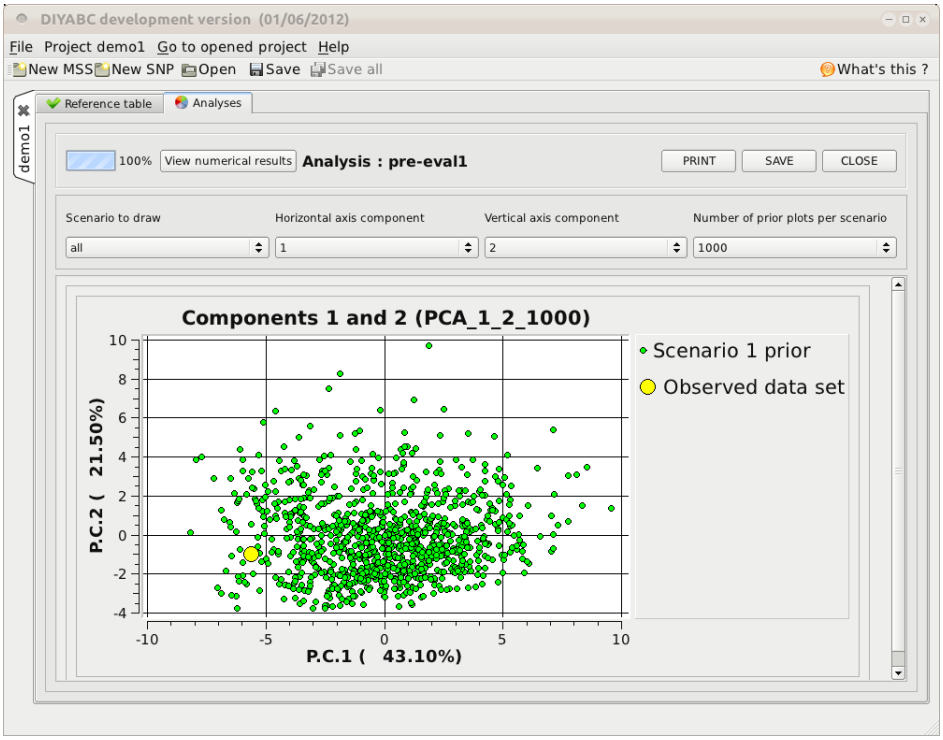
After clicking on the VALIDATE button, we go back to the previous screen. However, the new analysis now appears on top of the analysis panel. For each analysis, this panel provides its name and type, the list of parameters that will be transmitted (in a coded way) to the computation program, a progress bar that approximates the progress of the analysis run, and four buttons. The right button has to be clicked to launch the analysis. The three left buttons provide a way to copy an analysis (Copy button), to make some modifications (Edit button) before launching it or to delete the analysis (Del button).



Let's click on the Launch button. This analysis is very fast (ca 1 second) so that the progress bar shows almost immediately a 100% value :

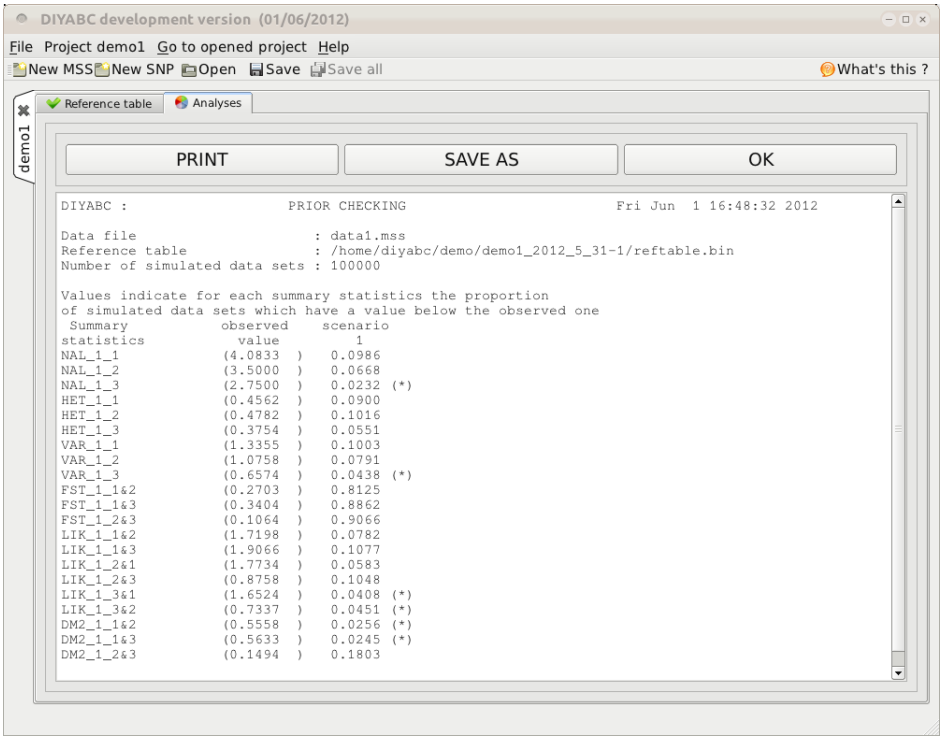


To view results, just click on the View results button. After some seconds (while the program reads the PCA result file), we can see this :



The results are shown PCA plane by PCA plane. Each (small) dot represents a simulated dataset from the reference table and the large yellow dot represents the observed data set. The initial components of datasets are the values of the summary statistics from which are computed the principal components. The four drop-lists (`Scenario to draw`, `Horizontal axis component`, `Vertical axis component`, `Number of prior plots per scenario`) can be used to explore further the results of the PCA.
The graphic can be printed or saved (PRINT and SAVE buttons, respectively). Clicking on the CLOSE button closes the result window. Eventually, clicking on the View numerical results opens up another

screen as shown below :



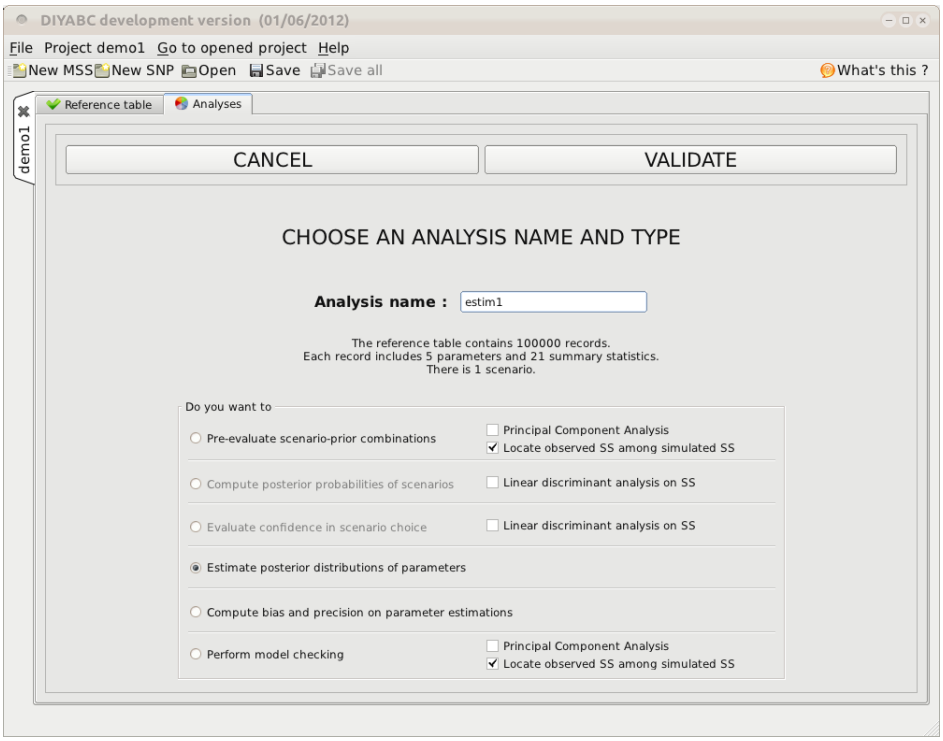
This screen is obtained by computing for each summary statistics the proportion of simulated data (considering the total reference table) that have a value below the value of the observed dataset. A star indicates proportions lower than 5% or greater than 95% (two stars, <1% or >1%; three stars, <0.1% or >0.1%).

As usual, results can be printed (PRINT) and/or saved (SAVE). Click on OK to leave this screen.

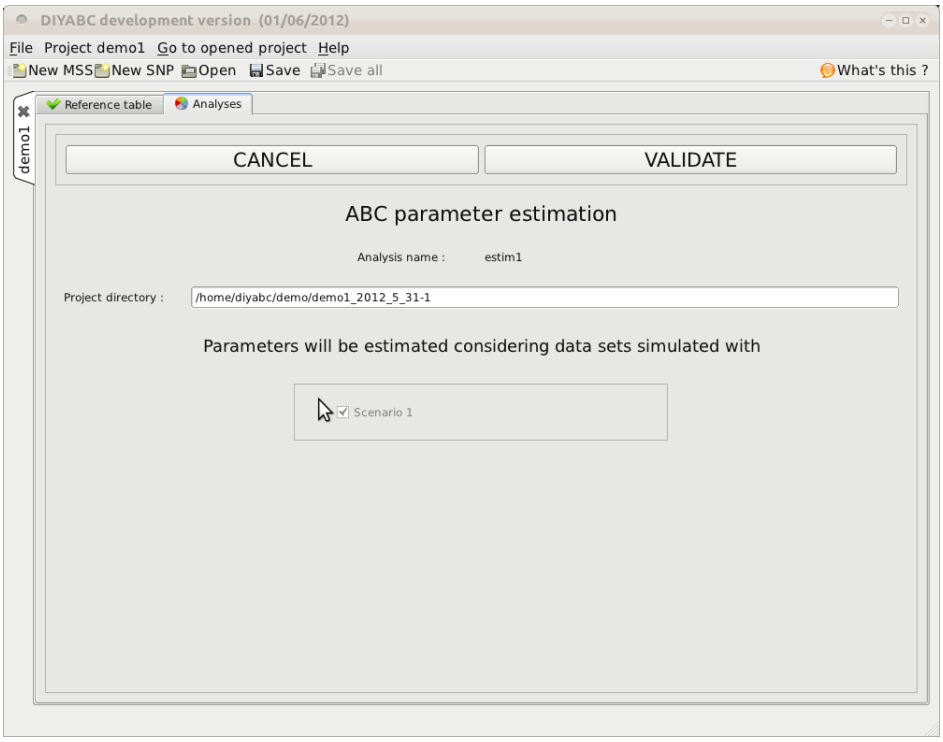
Although we get one star for a few summary statistics, we conclude that our model is suitable enough to proceed to other ABC analyses.

3.5.1 ABC parameter estimation

Back on the screen of page 30, we click on the Define new analysis button. We choose the `Estimate posterior distribution of parameters` option and we call `estim1` this second analysis :



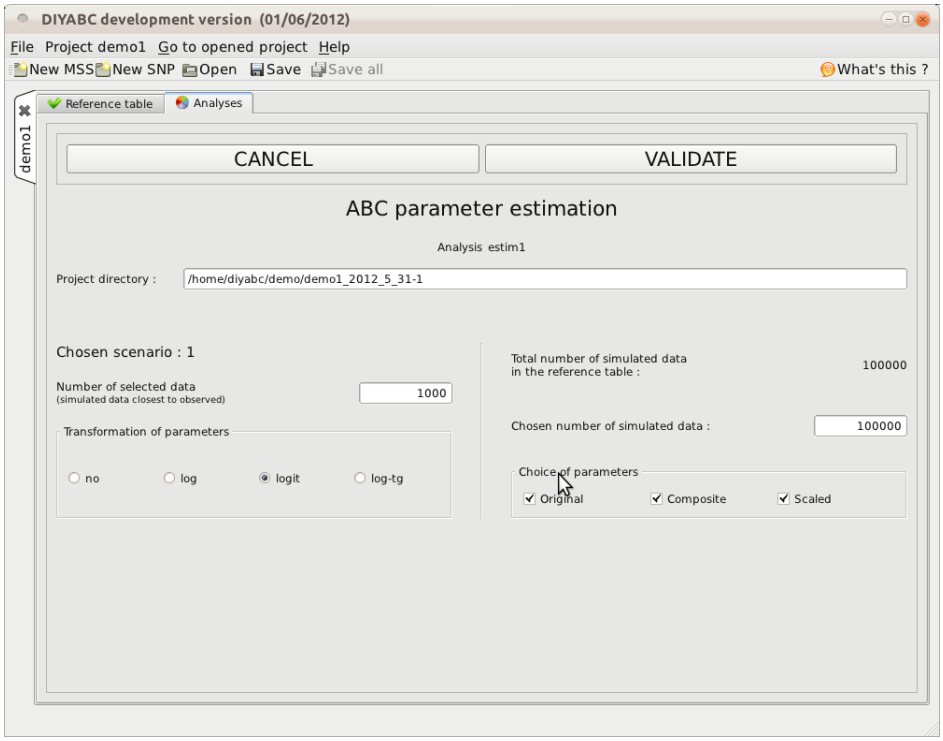
We click on the VALIDATE button and get the following screen in which we can choose the scenario to use for this estimation. Since a single scenario has been defined, there is nothing else to do than to click on the VALIDATE button :



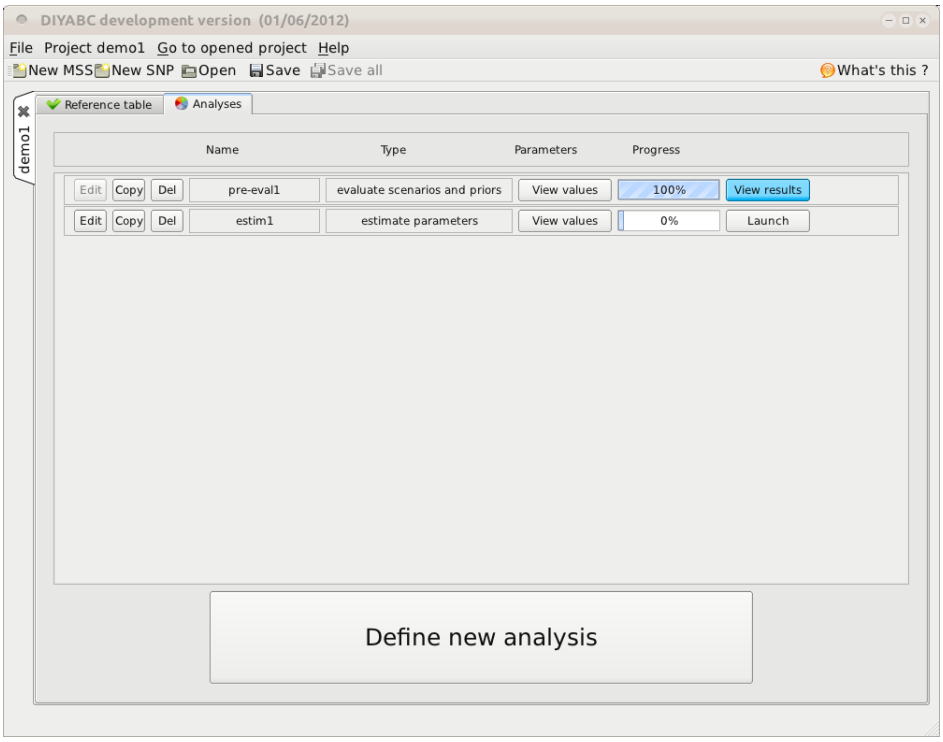
We get then the following screen in which we can make several choices :

- on the left hand side, we can choose the number of closest simulated datasets that will be used for the local linear regression (cf section 2.1).
- below, we can select the transformation of parameter values that can generally improve the results (default = logit transformation).

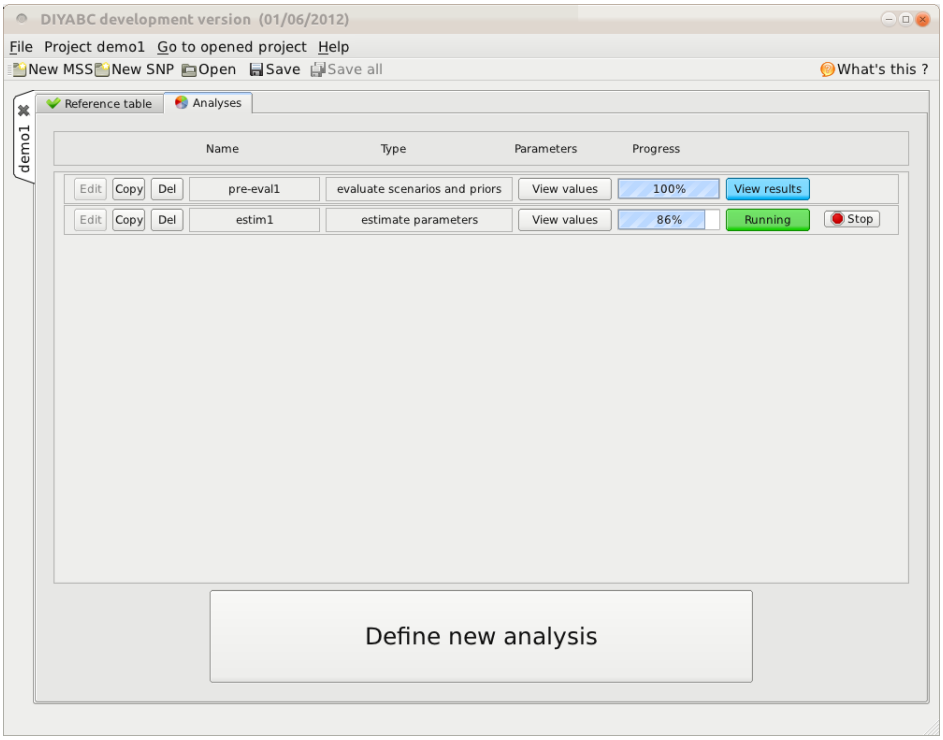
- on the right hand, we can truncate the reference table to a specified number of datasets.
- eventually, estimations can be performed either on original (*i.e.* raw) parameters, and/or combinations of parameters that are generally more estimable. *Composite* parameters are products of effective population sizes or times by mean mutation rate whereas *Scaled* parameters are ratios of effective population sizes or times by mean effective population size (computed from all terminal populations, *i.e.* N_1, N_2 and N_3 in the present example).



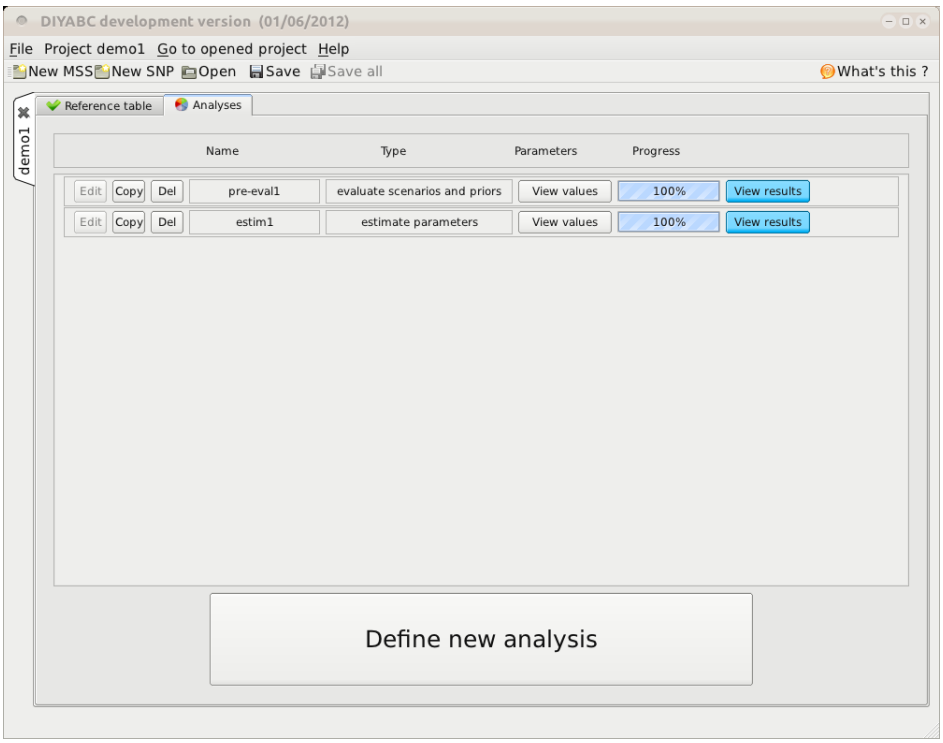
Apart from the number of closest datasets that we set at 10,000 (although 1,000 would be also a correct choice), we keep all other default values and click on the VALIDATE button. We get back to the Analysis control panel which now looks like this:



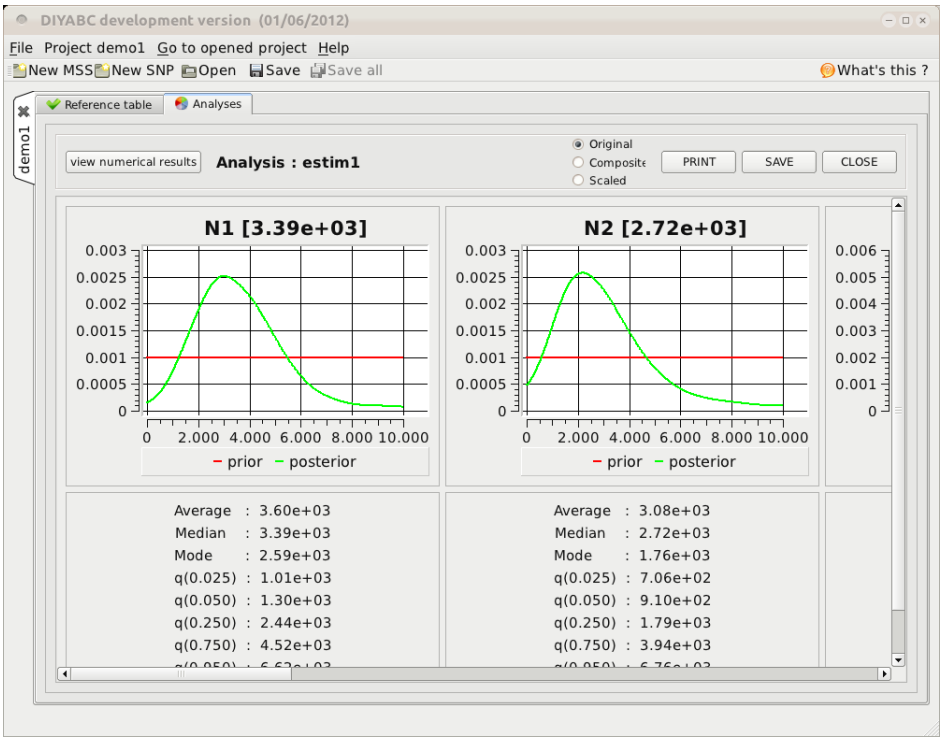
We click on the Launch button. The analysis progress is now visible :



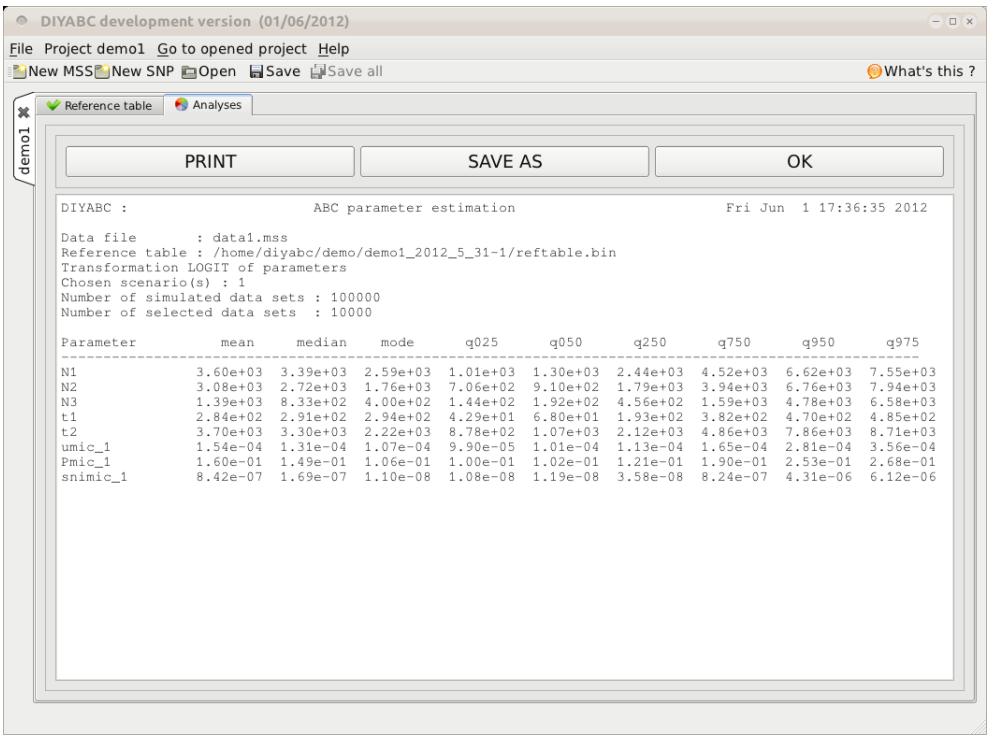
As long as the analysis is not terminated, we could stop it by clicking on the Stop button. Once this second analysis is finished, we can view its results by clicking on the View results button :



Let's have a look :

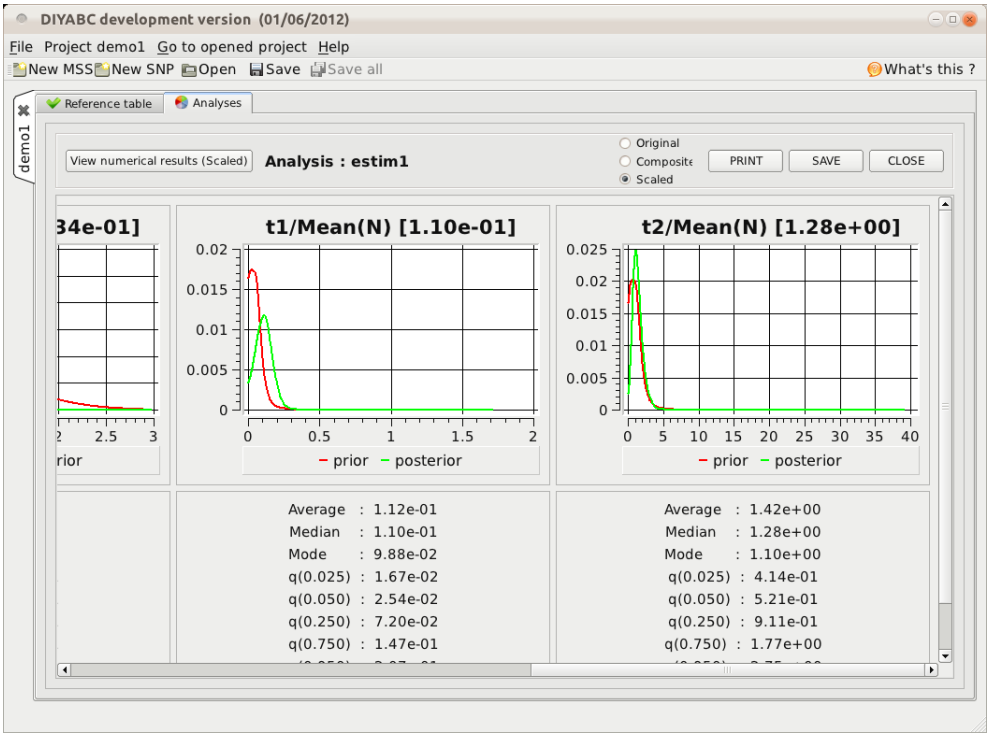


In the scrolling window, we get graphics showing the prior (red curve) and posterior (green curve) distributions of all parameters. Below each graphics are statistics (mean, median, mode and quantiles) of the posterior distribution. The latter are grouped in a table that appears when clicking on the upper left view numerical results button, showing this :



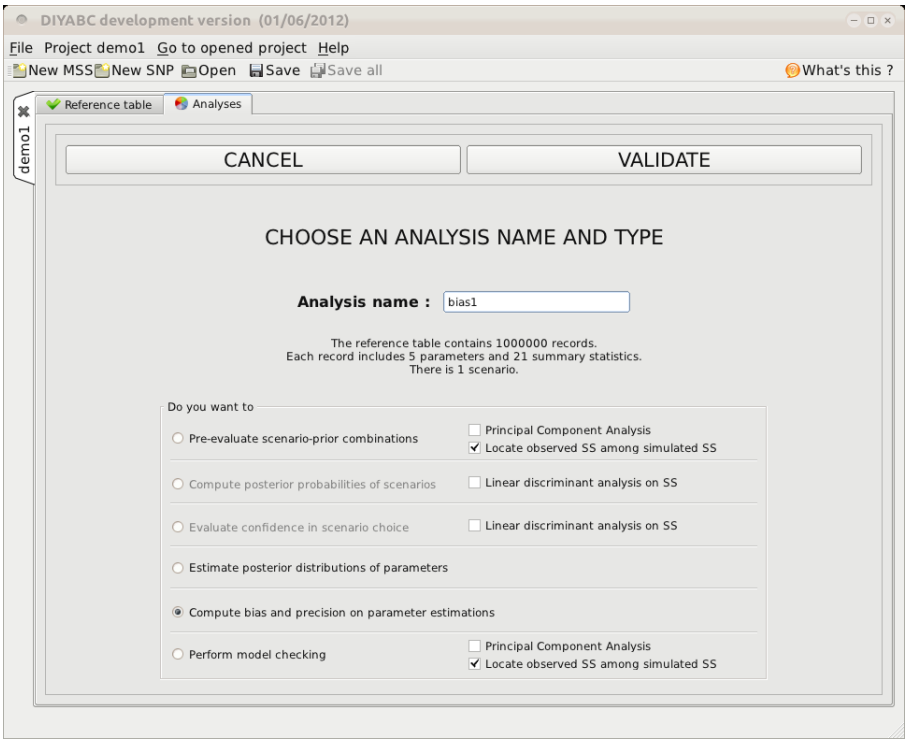
We go back to the previous screen by cliking the OK button.

We can also have results for *Composite* or *Scaled* parameters. Below is an example of *Scaled* time parameters obtained by clicking on the *Scaled* radio button and scrolling the graphs window to the right:



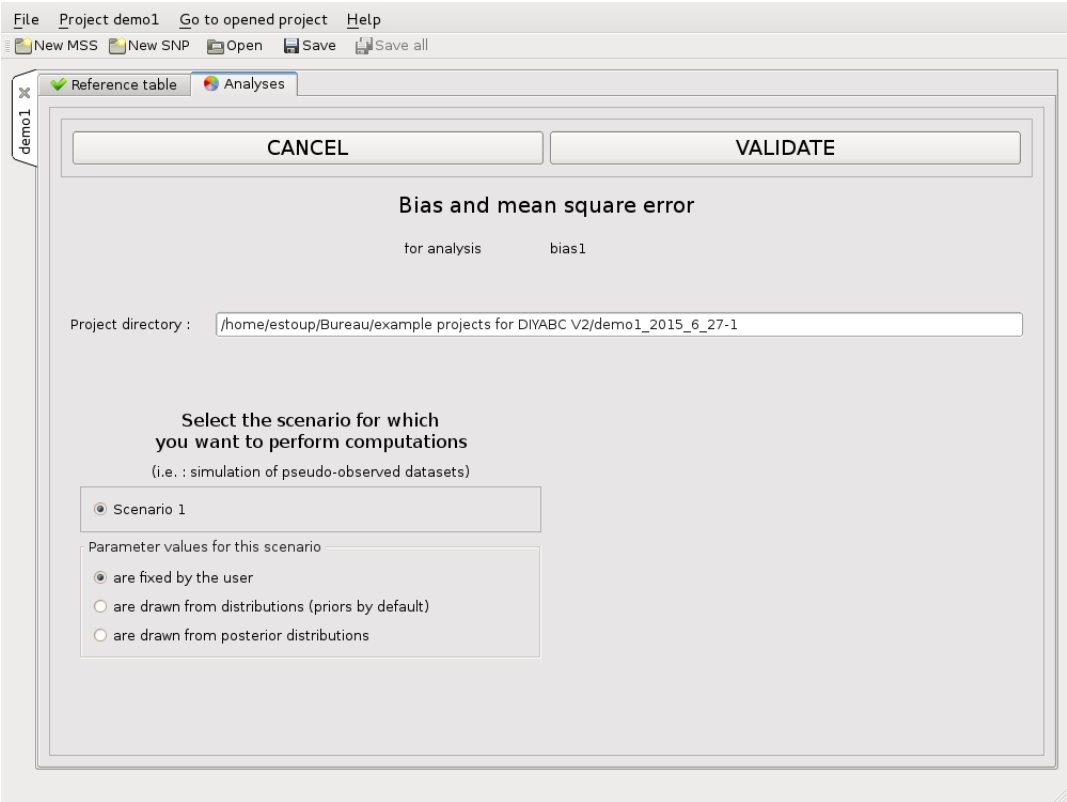
3.5.2 Bias and precision

Let's define a new analysis (click on the Define new analysis button) and choose the option `Compute bias and precision on parameter estimations`. We give it the name `bias1` :



In this kind of analysis, peudo-observed datasets are simulated with known values of parameters copy-

ing the exact configuration of the observed dataset in terms of sample sizes (taking into account missing data) and are submitted to the same ABC estimation process. If we assume that the evolutionary scenario is correct, the comparison of real and estimated values of parameters provide some information of the precision on the estimation process.
We validate and get this screen :



The demographic, historical and mutational parameters values of the pseudo-observed-datasets (pods) can be produced from a single scenario (here scenario 1 which is the only one avalaible in the present analysis) in three different ways:

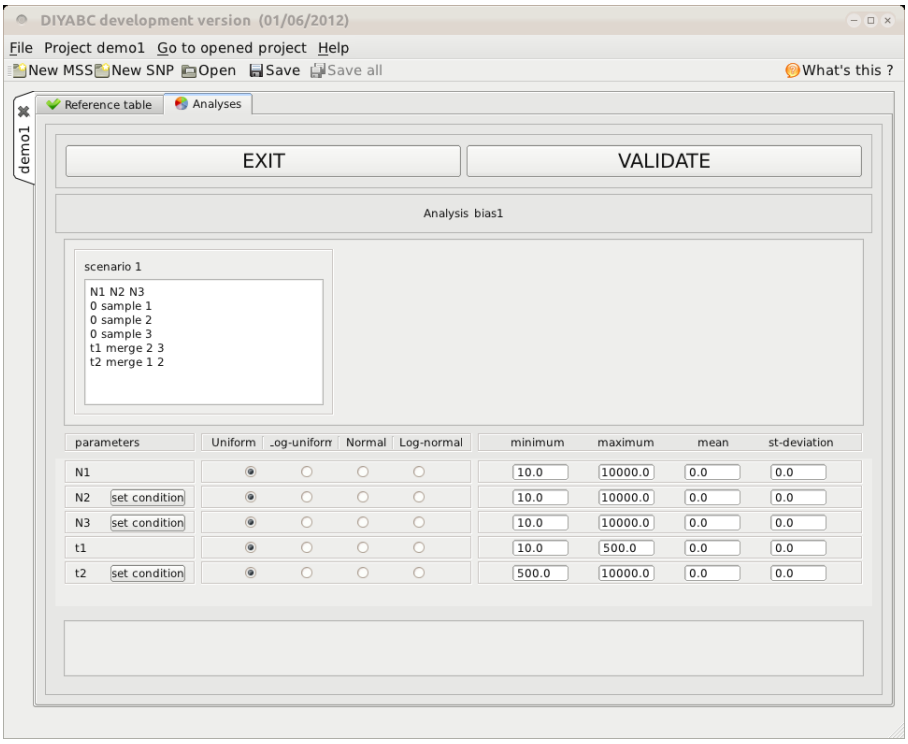
(i) they correspond to a set of fixed values chosen by the user;

(ii) they are drawn drom the initial prior distributions (which can be modified by the user);

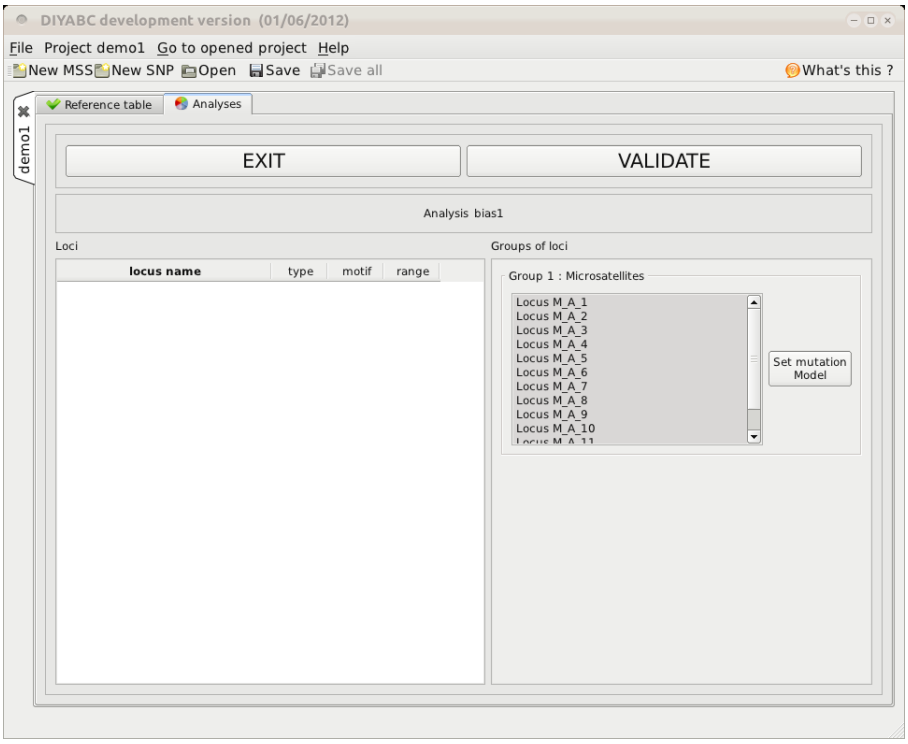
(iii) they are drawn from the parameter posterior distributions estimated using a standard ABC procedure. Note that computing accuracy indicators conditionally to the observed dataset (i.e. focusing around the observed dataset by using the posterior distributions) provide a more relevant estimation of accuracy of parameter estimation in the vicinity of the observed dataset (which is the location of prime interest in the vast data space defined by prior distributions) than blindly computing accuracy indicator over the whole prior space.

We first choose to draw parameter values from prior distributions by clicking on the option “are drawn from distributions (prior by default)” and on the VALIDATE button:

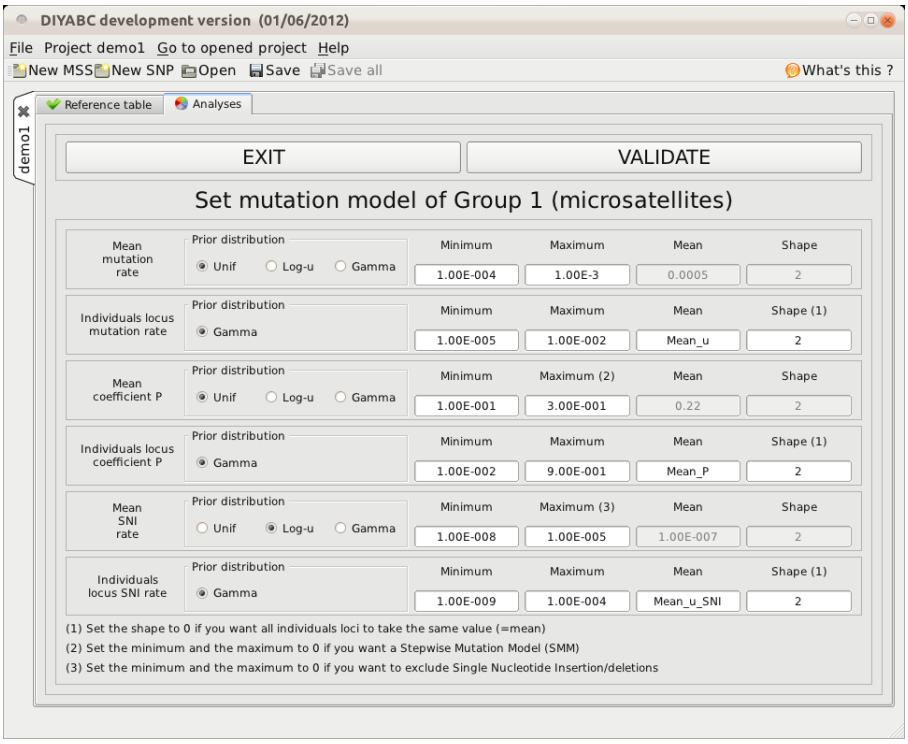
We get this screen which allows us to choose distributions for demographic and historical parameters.



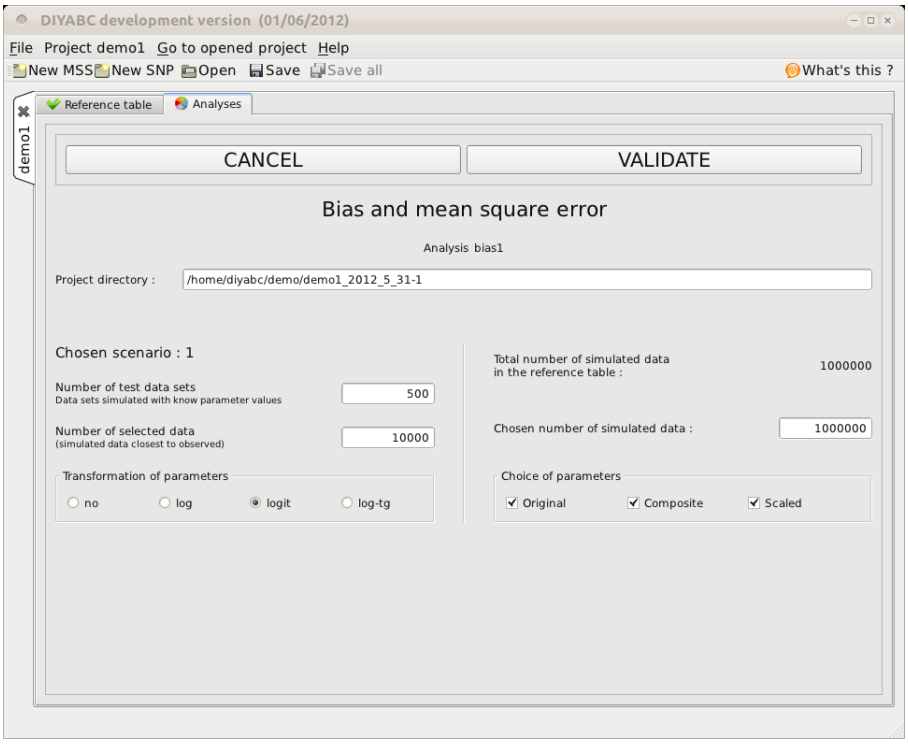
By default, the following screens suggest the prior distributions that have been used to build the reference table. However, these distributions can be edited if necessary. We decide not to change them and click on VALIDATE which brings us to the following screen :



If we want to keep the same distributions for mutation parameters as when builduing the reference table, we just click on VALIDATE. If we need to change them, we click on Set mutation model which would bring the following screen :

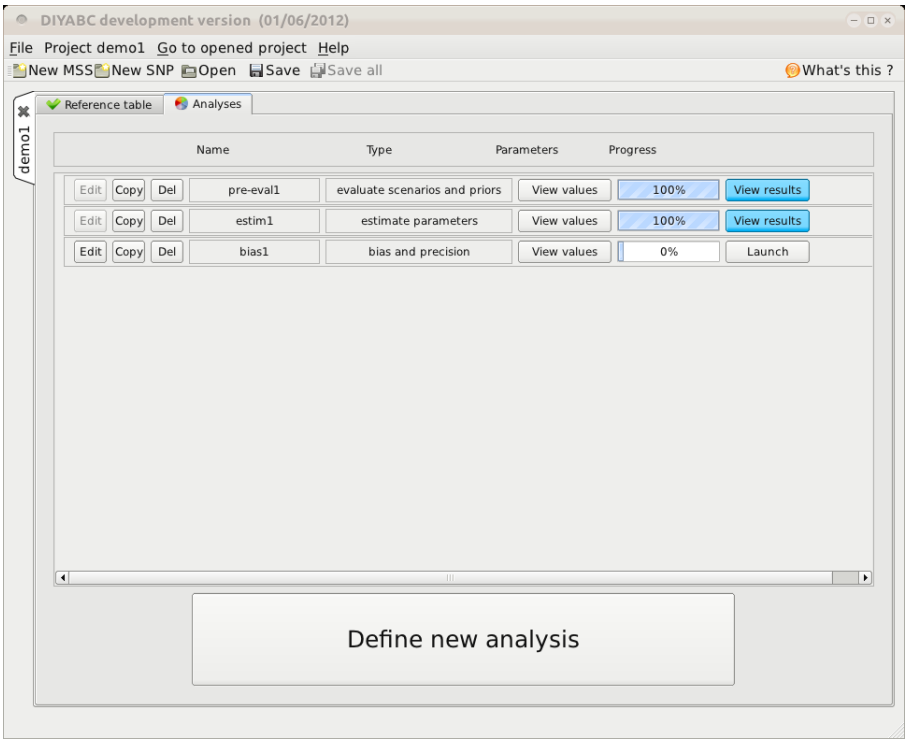


After validating twice, we get the last screen necessary to define this kind of analysis :

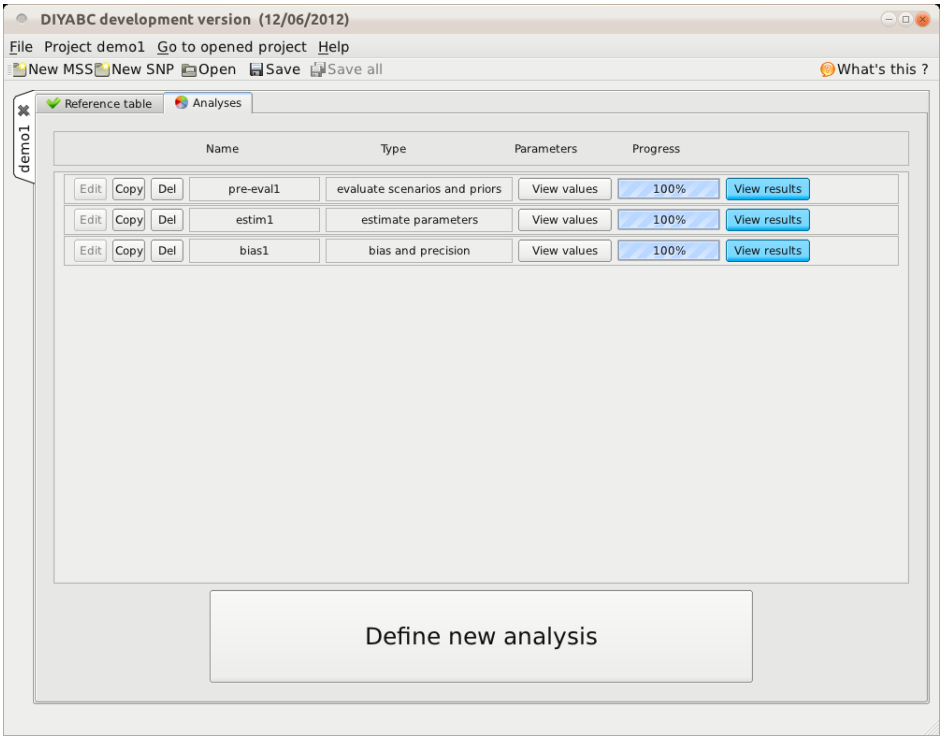


This screen is similar to that for parameter estimation (see section 3.5.1). The proposed (and potentially modifiable) parameters "number of selected data" and "Chosen number of simulated data" are those that will be used to proceed the ABC parameter estimations for each pod (with parameter values drawn from prior distributions). The default number of pods (i.e. test data sets) is 500 but it can be increased to e.g. 5,000 for a more precise estimations of the accuracy measures.

After validating, we get back to the analysis panel with a third analysis defined :

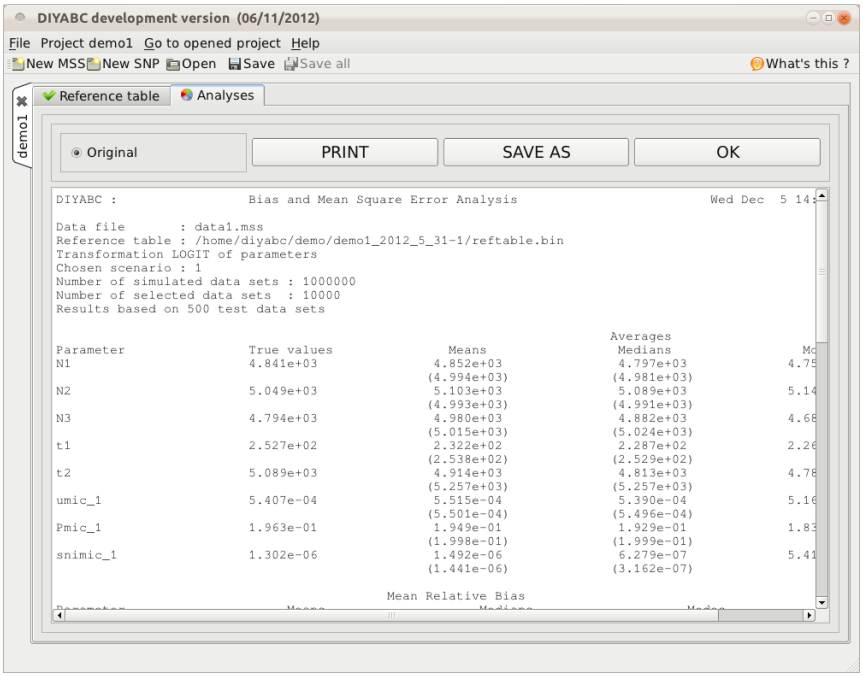


The analysis takes some time to run compared to the previous one, because it simulates hundreds datasets and on each one, a full ABC estimation is performed. Then after some time, the analysis is finished:



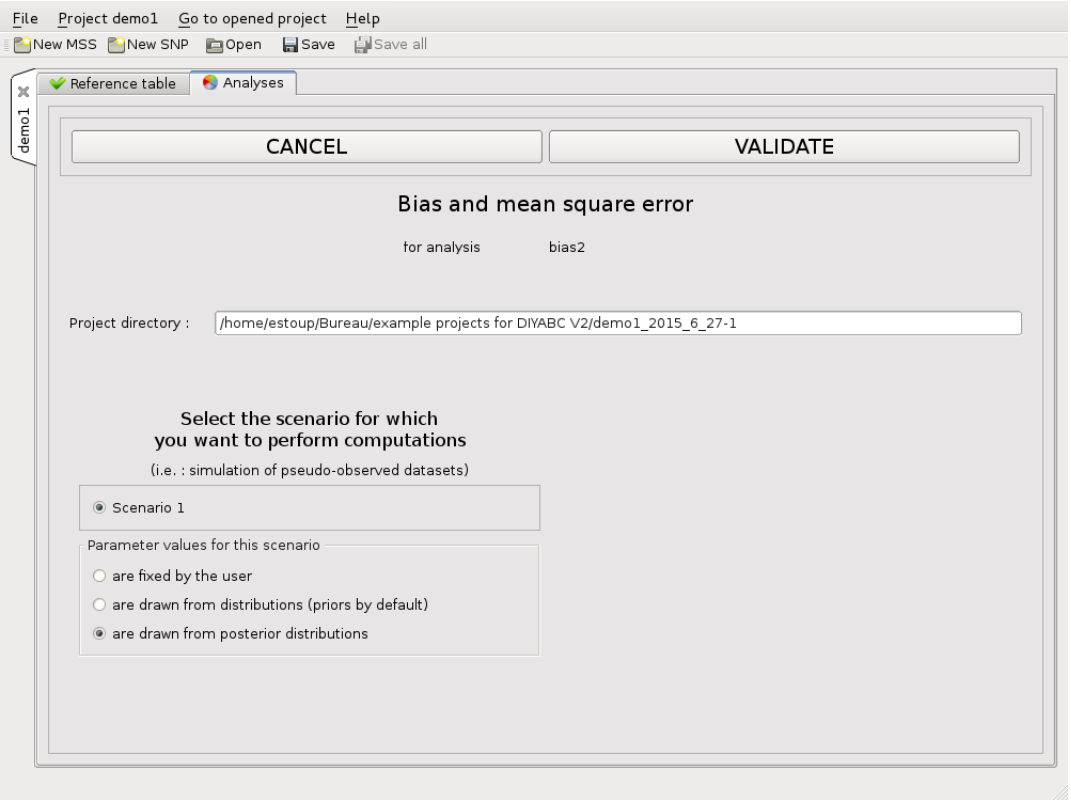
To view results, we click on the View results button.

The results are visible in a scrolling window :

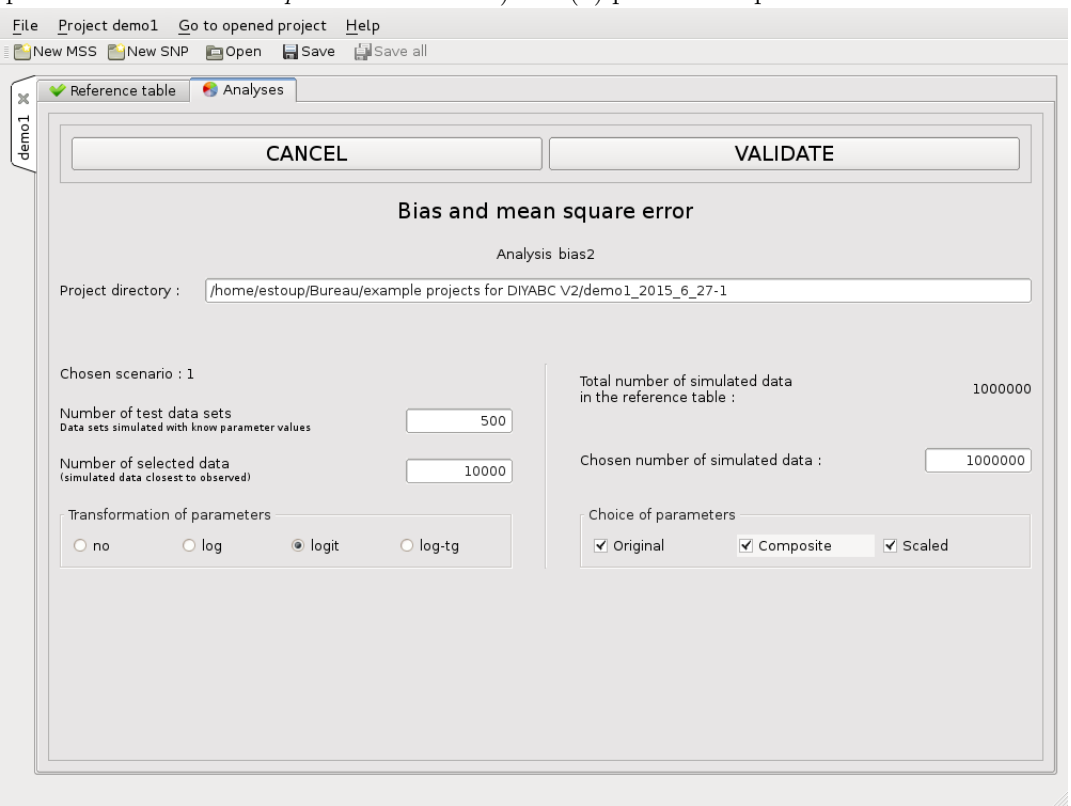


Accuracy measures are available for original, composite and scaled parameters. Note that there are two values given for each accuracy measures. The upper value is that of the statistics computed from the *posterior* distribution of parameters, *i.e.* **using the genetic information provided by data**. The lower value, noted between parentheses, is that of the statistics computed from the *prior* distribution of parameters, *i.e.* **NOT using the genetic information provided by data but only that contained in prior distributions**. The output file includes various measures of accuracy such as those detailed in section 2.11. Smaller accuracy values (e.g. small RMSE or RMedAD values) correspond to more precise parameter estimations. Each accuracy measure is associated with a second value given between parentheses corresponding to the accuracy measure without taking into account the genetic information provided by the data. The comparison of the two values provides a rough assessment of the amount of information provided by the genetic data in the inferential process. Note that some accuracy values computed using only prior information may be (surprisingly at first sight) smaller than accuracy values taking into account genetic data: this may occurs when the genetic data contain little information and produce systematic estimation biases.

We then choose to run a new analysis of the same type but *this time drawing parameter values from posterior distributions*. We click (again) on the Define new analysis button) and choose (again) the option `Compute bias and precision on parameter estimations`. We give the name `bias2` to this new analyis. After validating this new analysis, we click on the option "are drawn from posterior distributions" and on the VALIDATE button.

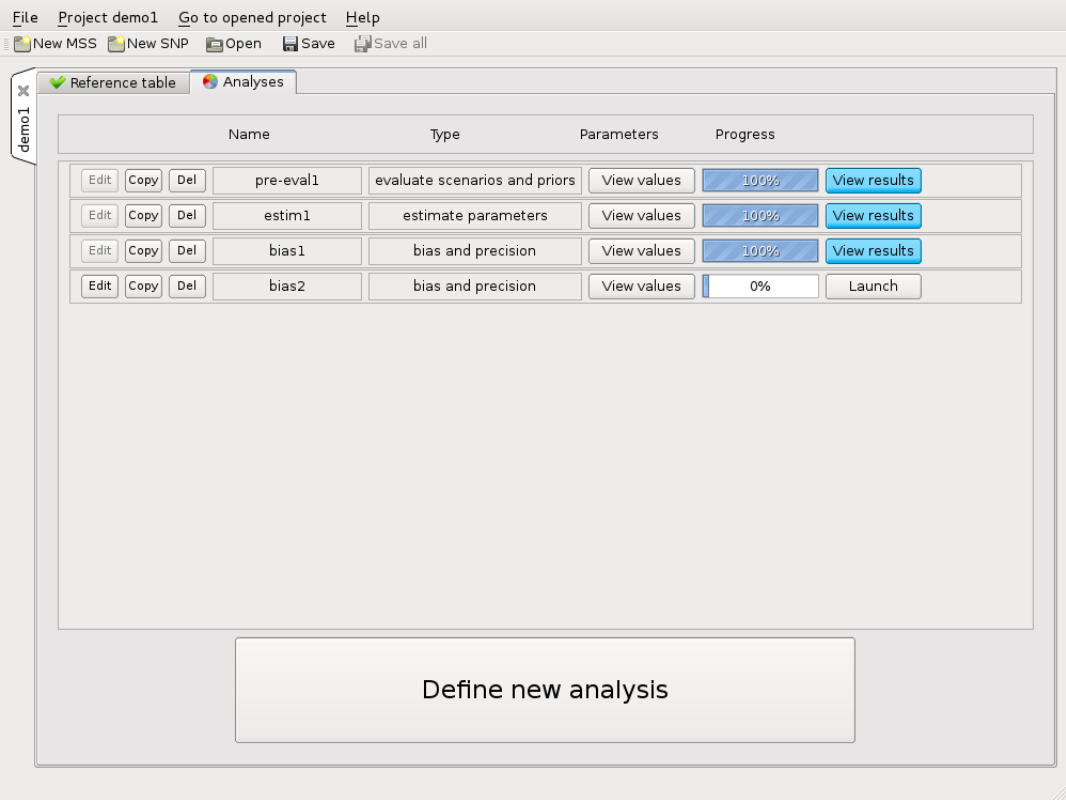


The screen immediately following is similar to that for parameter estimation (see section 3.5.1). The proposed (and potentially modifiable) parameters "number of selected data" and "Chosen number of simulated data" are those that will be used to both (i) make in a first phase an estimation of the parameter posterior distributions of the *observed dataset* (hence defining the distributions from which the parameter values of the *pods* will be drawn) and (ii) proceed the parameter estimations for each pod.

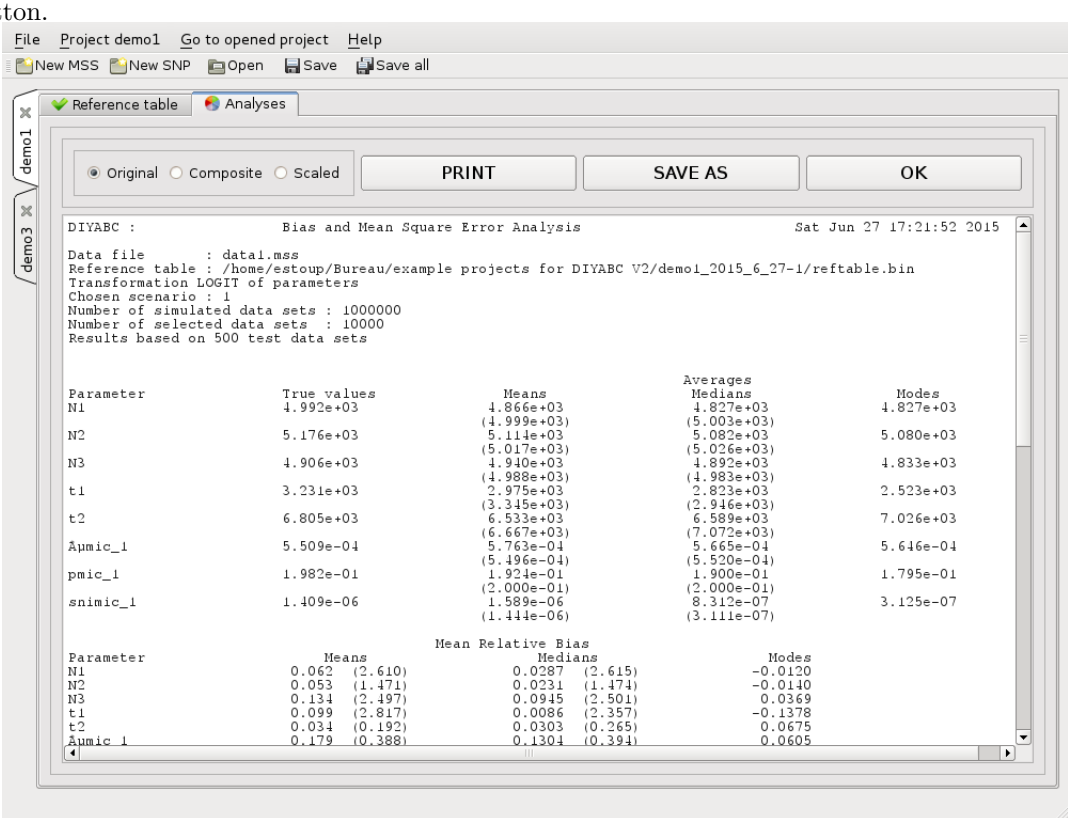


The default number of pods (i.e. test data sets with parameter values drawn in this case from posterior distributions) is 500 but it can be increased to e.g. 5,000 for a more precise estimations of the accuracy measures.

After validating, we get back to the analysis panel with a third analysis defined (named bias2).



As for bias1 analysis, the bias2 analysis takes some time to run because it simulates hundreds test datasets (usually between 500 and 5,000 pods) and on each one, a full ABC estimation is performed. Then after some time, the analysis is finished and one can view results by clicking click on the View results button.

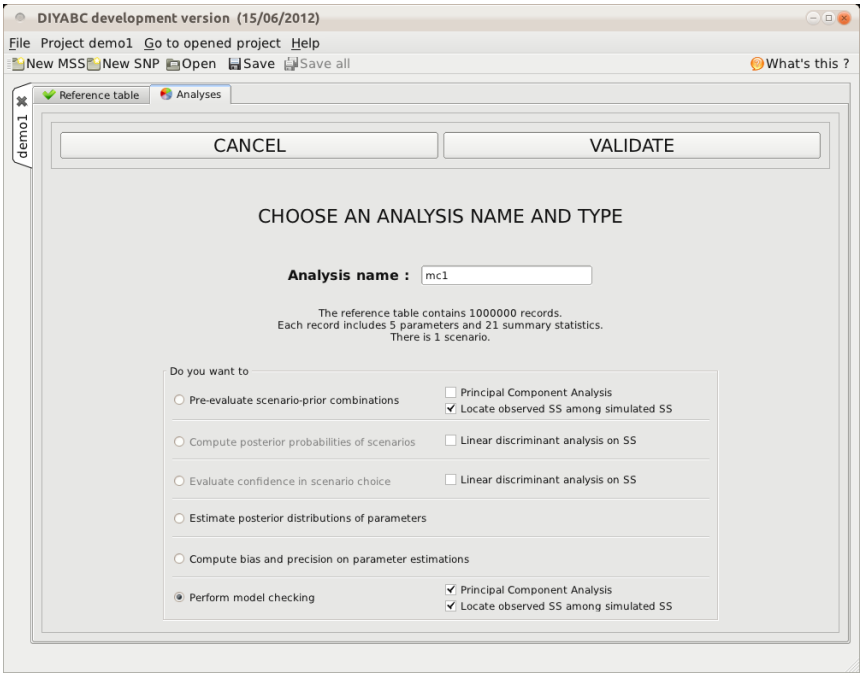


Outputs are similar to those of the bias1 analysis except that the pod's parameters have been drawn from the posterior distributions of the observed dataset. The bias2 estimation hence provides a more relevant estimation of accuracy of parameter estimation in the vicinity of the observed dataset than

blindly computing accuracy indicator over the whole prior space as in the bias1 analysis. As for the bias1 analysis, the accuracy measures are available for original, composite and scaled parameters.

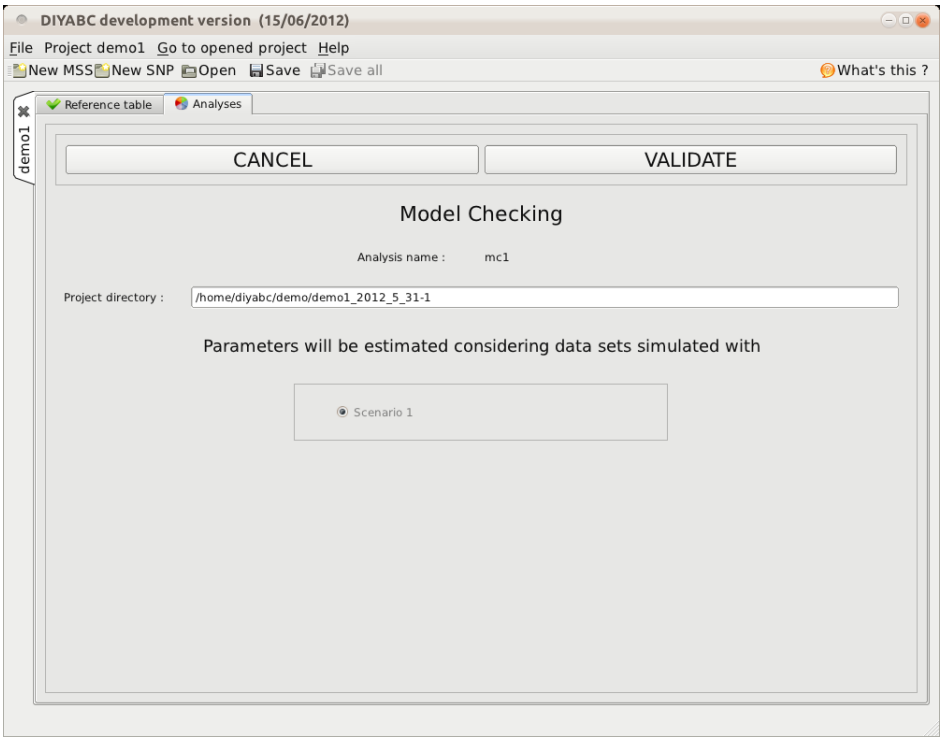
3.5.3 Model Checking

We now define another type of analysis called `Model Checking` which is used to evaluate how well the scenario and priors of parameters fit the data summarized by summary statistics. This is the last option on the following screen :

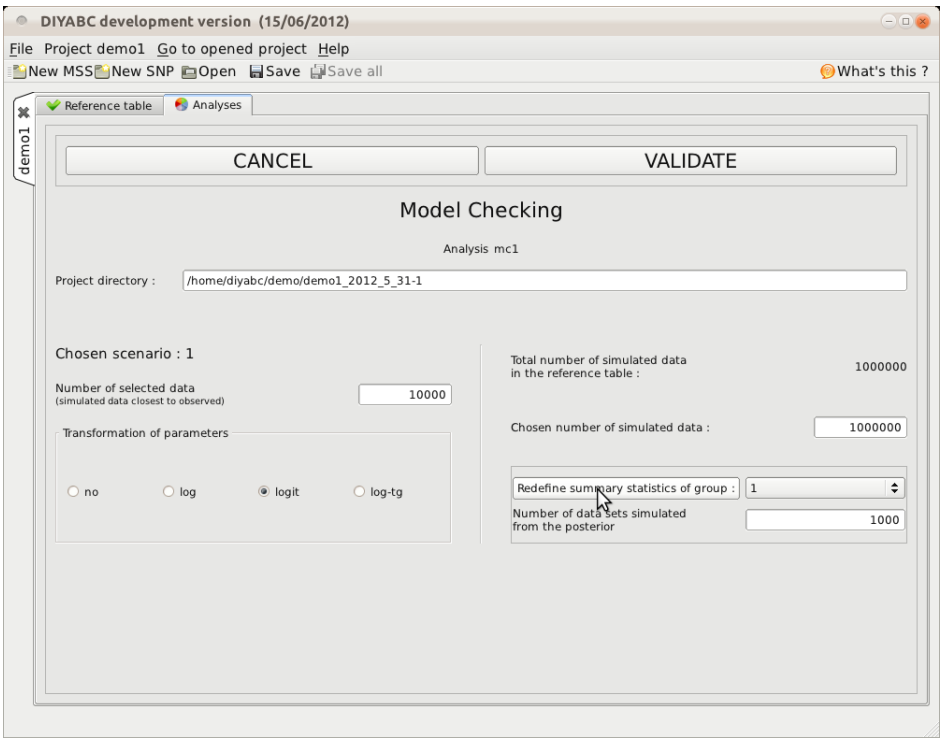


We call this analysis `mc1` and check the box to get a PCA performed. This PCA is computed in the same way compared to that of the first option (`Pre-evaluate scenario prior combinations`). However, new datasets simulated with parameters drawn from the posterior distributions of parameters are also represented on the different planes of the PCA (but not taken in the PCA computation).

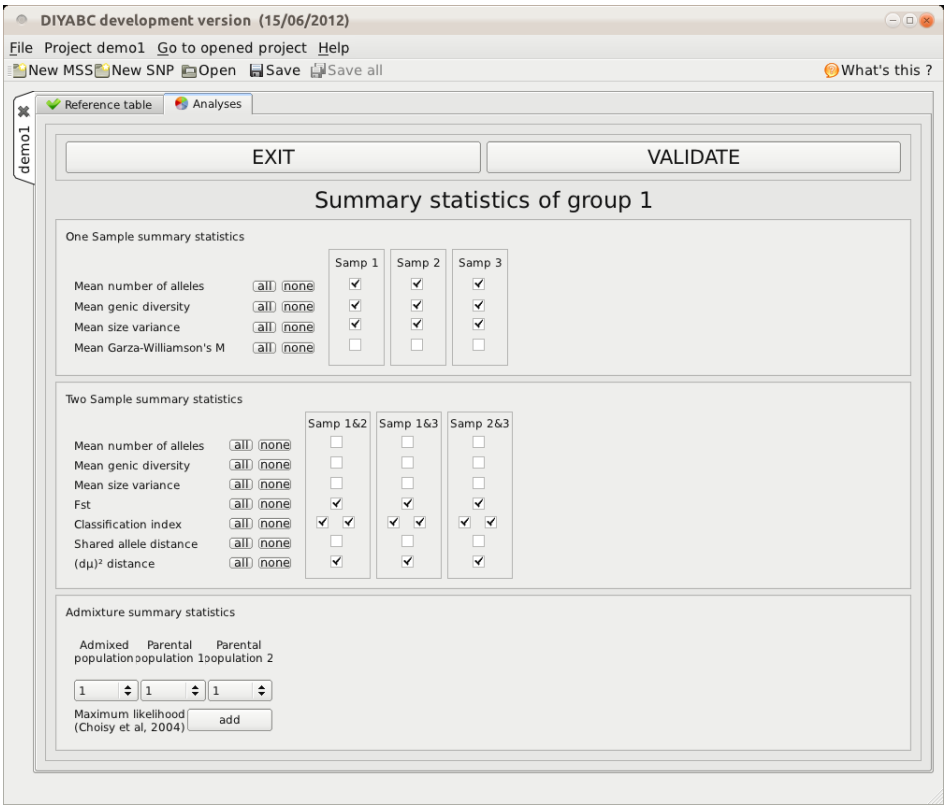
We validate the above screen and get the usual next screen :



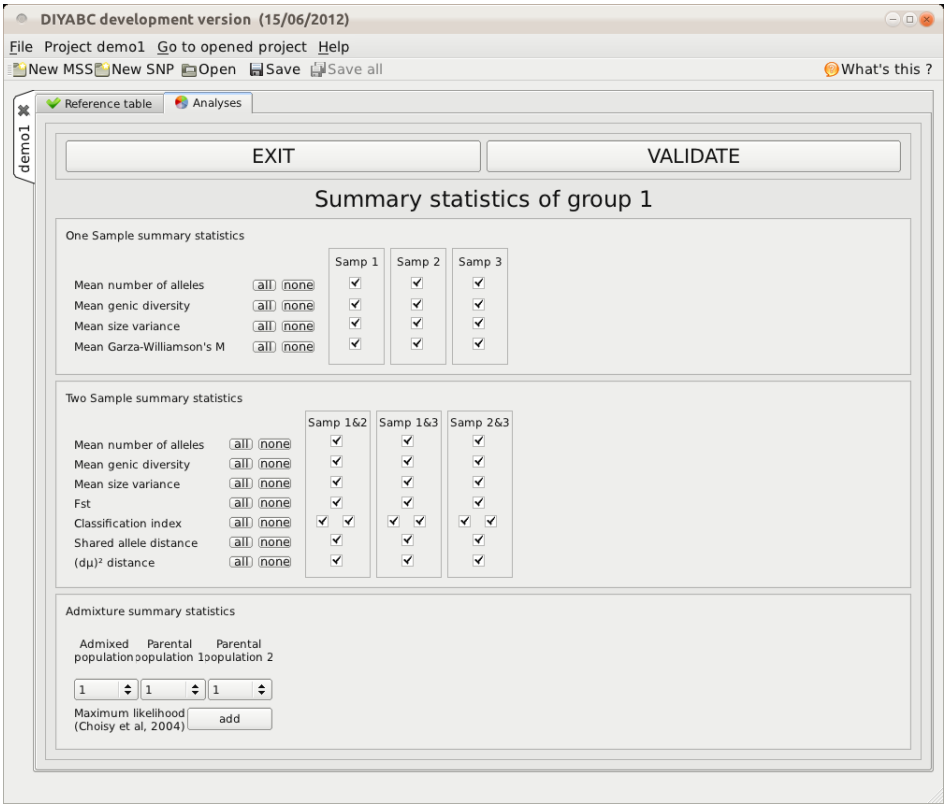
that we just validate to get the following screen :



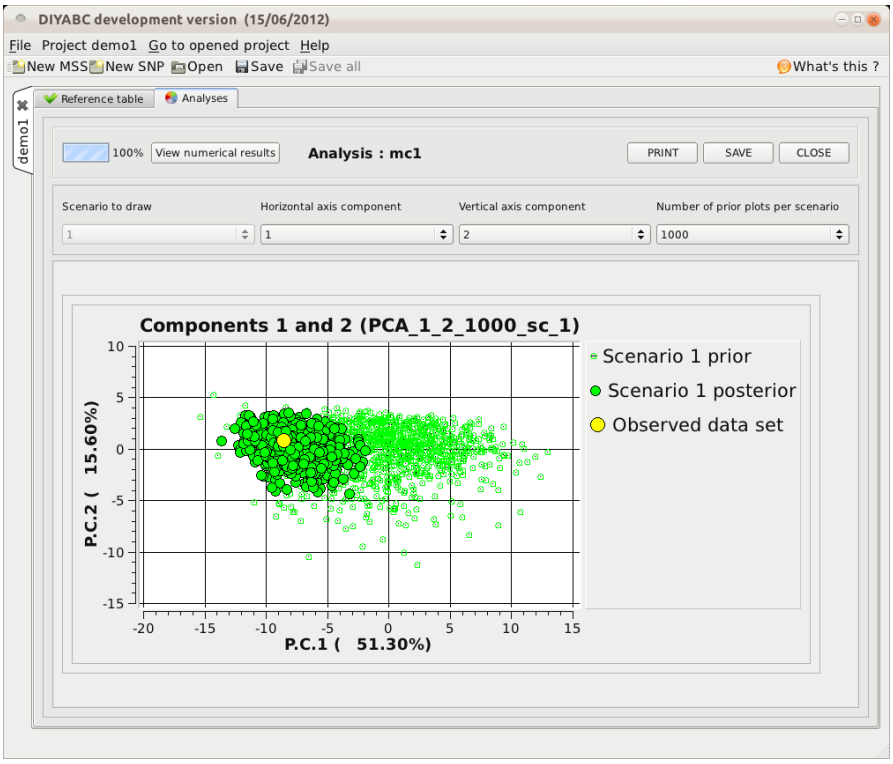
In this screen which we have already seen, there is a new panel (bottom right) in which we can choose the number of datasets that we want to simulate from the posterior distributions of parameters. There is also a button Redefine summary statistics of group: shown by the pointer. This button allows to change the set of summary statistics (for a given group of loci chosen through the drop list on the right). Clicking on this button opens up the usual following screen in which, by default, are checked the summary statistics in the reference table.



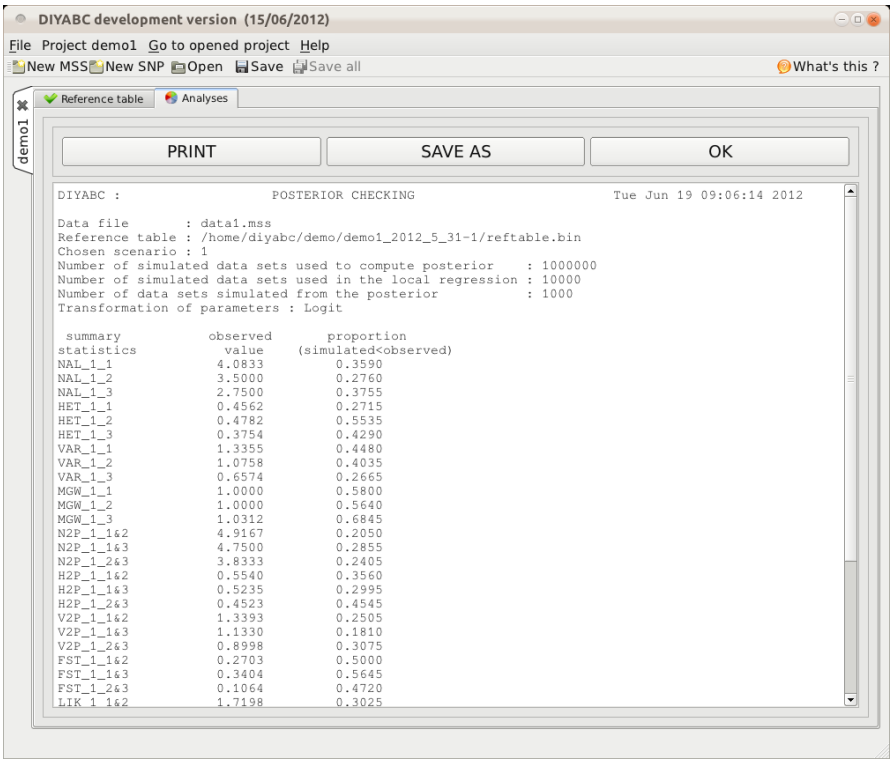
We decide to use all one-sample and two-sample summary stats :



Note that when the set of summary statistics is changed (as here), it is necessary to also simulate a large number of datasets using parameter priors to get corresponding values of the newly introduced summary statistics. We validate twice and launch the analysis. When it is finished, we click on the View results button and get this screen:



Clicking on the View numerical results leads to the following screen which provides, for each individual summary statistics, the value in the observed dataset as well as the proportion of data sets (simulated from the posterior) that have a value lower than the observed data set.



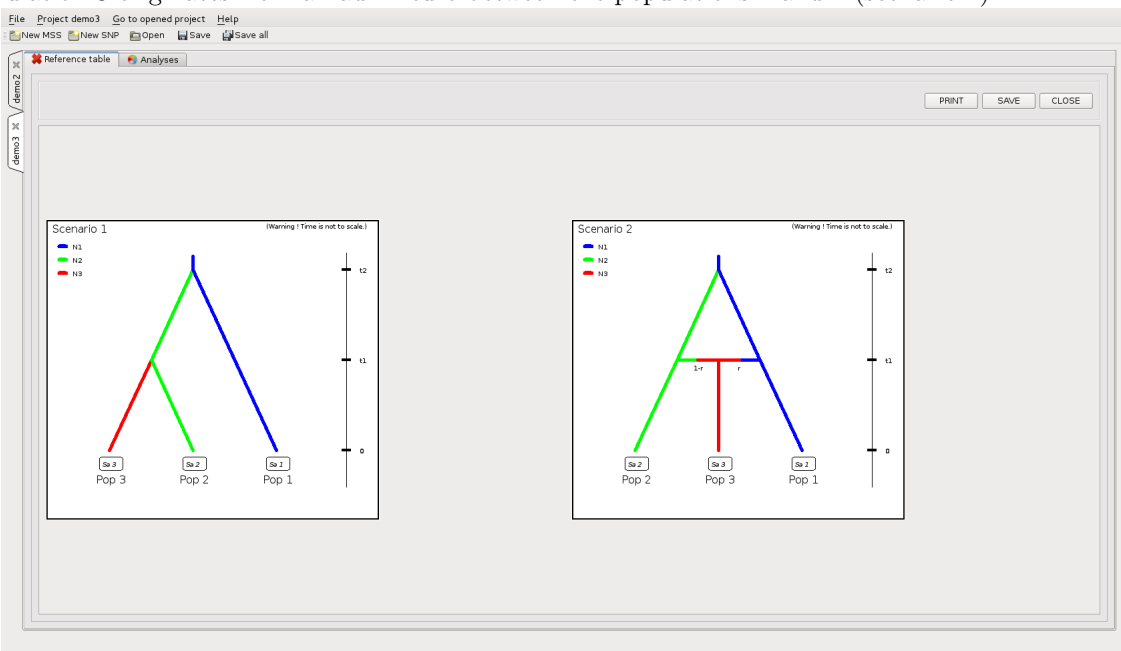
Notice that in this computation, values that are in the interval $[s_{obs} - 0.001, s_{obs} + 0.001]$ are counted for one half those that are outside the interval. This explains why the fourth digit of the proportion can be 0 or 5 while having simulated 1000 data sets.

Here the conclusion is that the chosen model/posterior explain correctly the observed dataset (see Cornuet

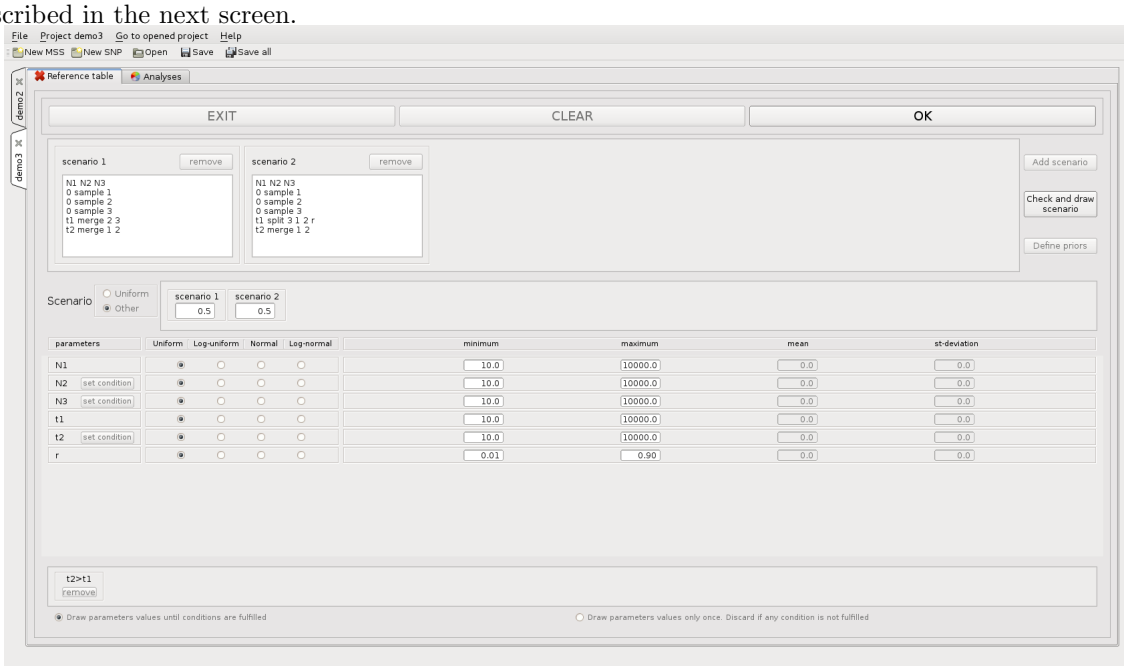
et al. (2010) for further illustrations).

3.5.4 Posterior probabilities of scenarios

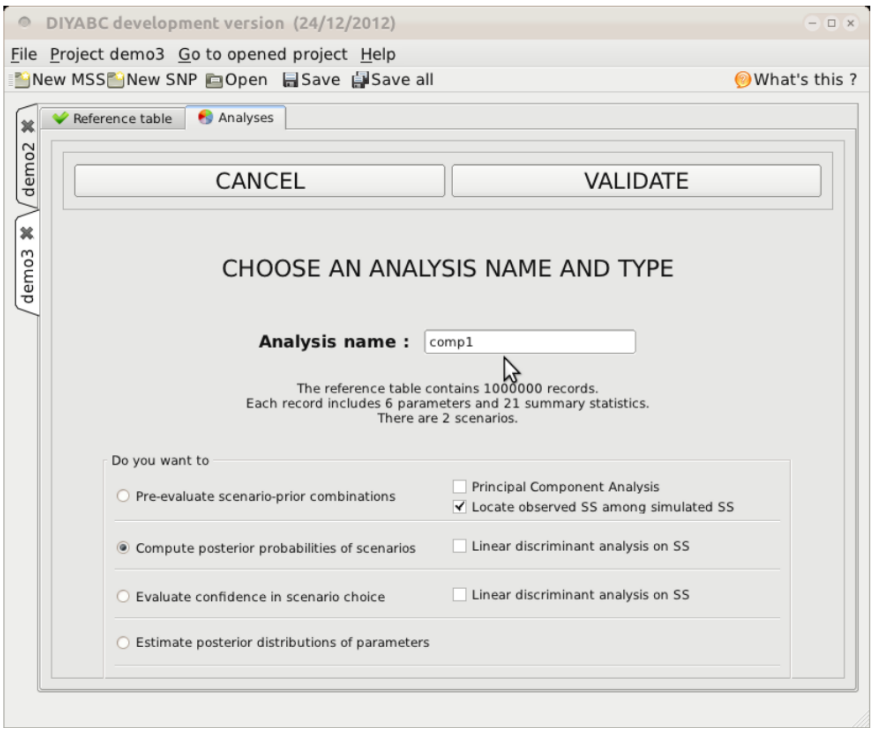
Consider a new example dataset in which three populations have been sampled. We want to decide which scenario is the best supported by data, a divergence scenario (scenario 1) or a split scenario in which the population 3 originates from an admixture between the populations 1 and 2 (scenario 2) :



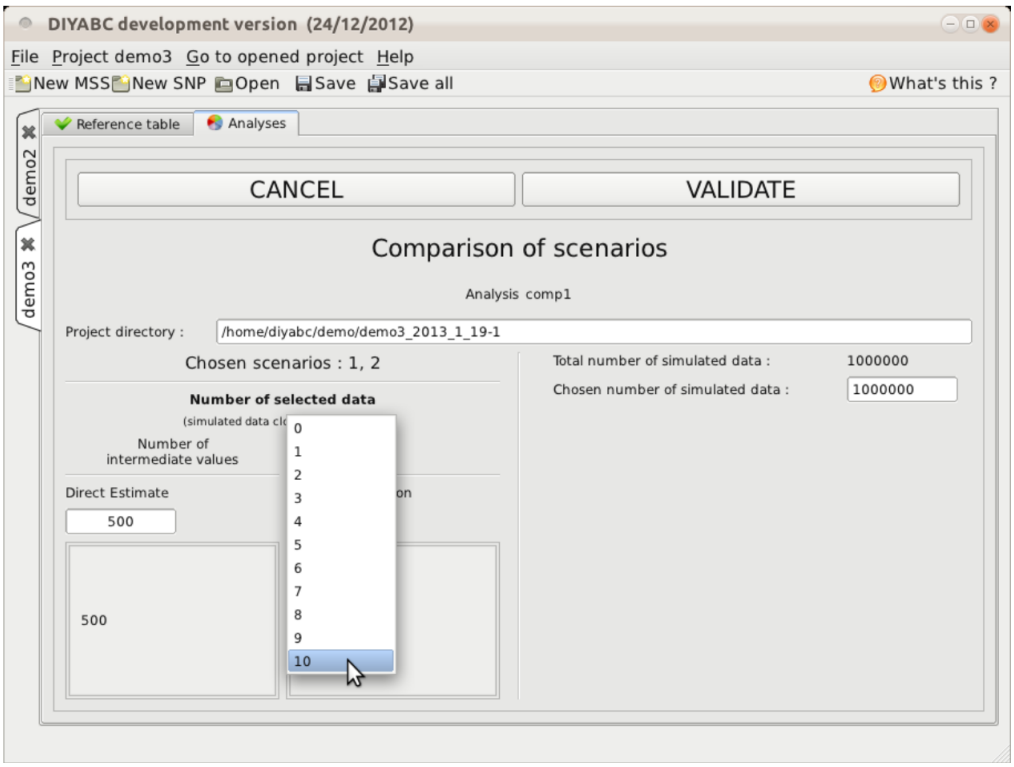
We first built a reference table with 1,000,000 simulated datasets (500,000 for each scenario) summarized with the same statistics as above and drawing parameter values into the prior distributions described in the next screen.



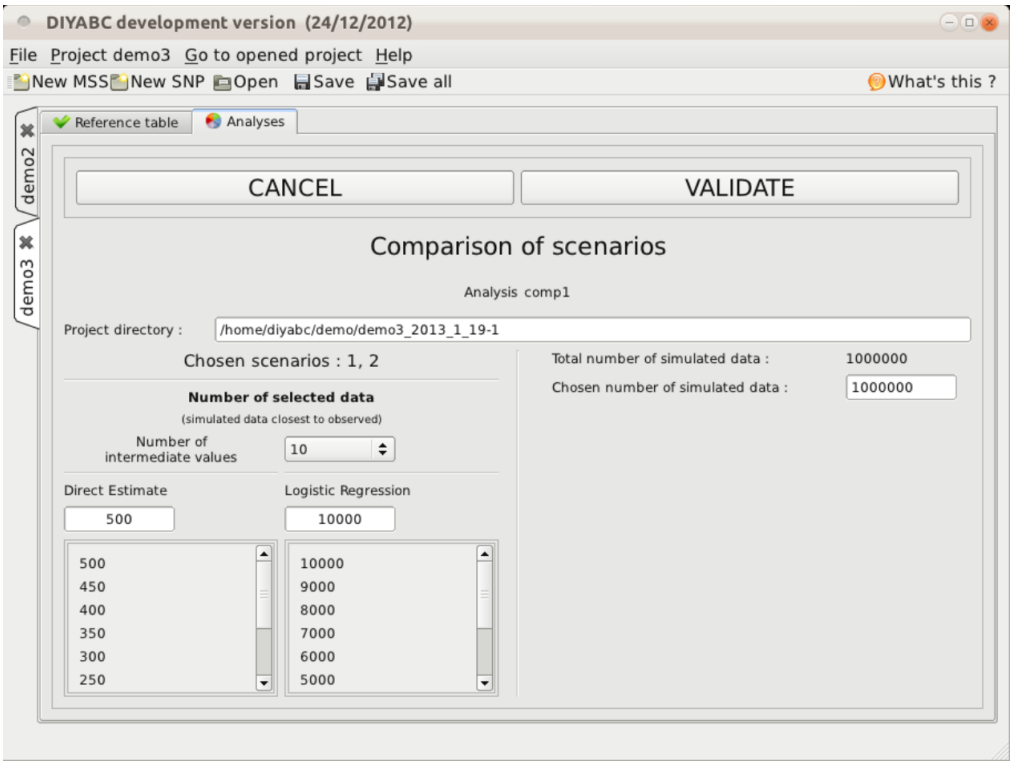
We then define a new analysis that we call `comp1` :



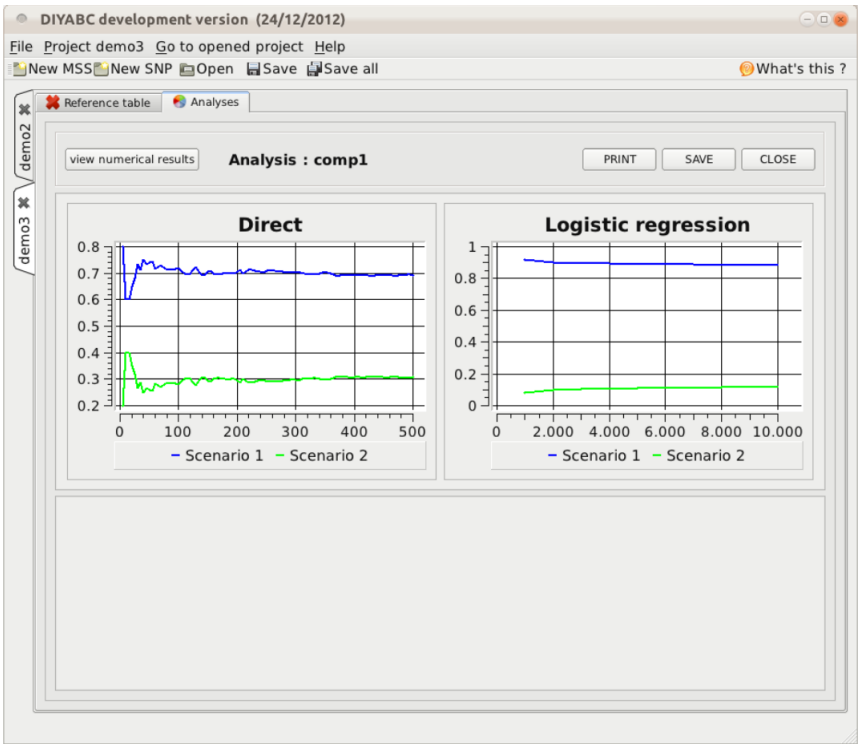
It is worth stressing here that it is possible to *replace original summary statistics (SS) by discriminant scores by checking the box* `Linear discriminant analysis on SS`. This option is useful when there are numerous scenarios and many summary statistics (see Estoup et al. 2012). However, in the present case, this is not necessary since the analysis with a relatively few original summary statistics and only two scenarios to be compared takes only a few seconds. After clicking on the VALIDATE button, we fill in the required fields, taking default values except for the number of local linear regression (on the second screen) that we set to 10:



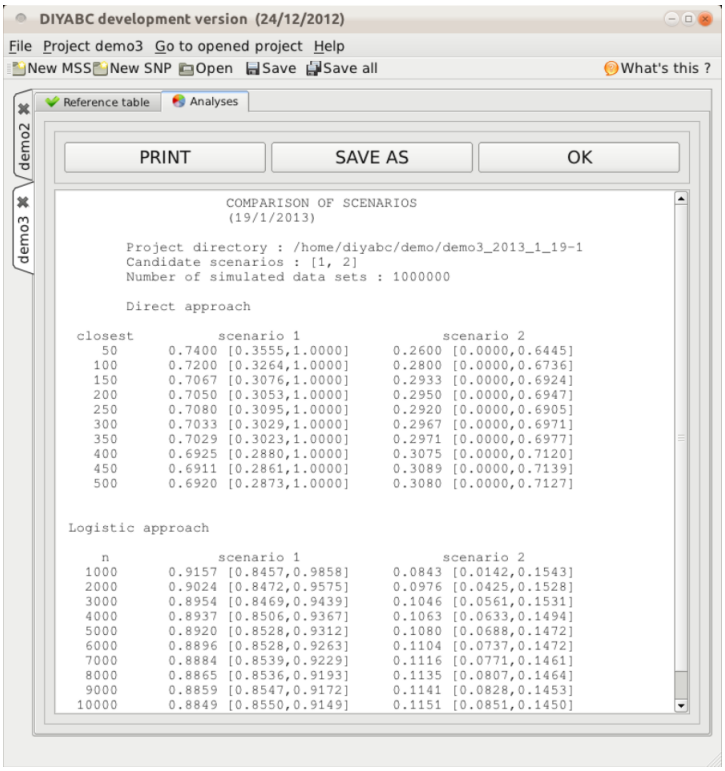
So that we get the following screen :



After validating the screen above, we launch the analysis which lasts a few seconds and press on the View results button. The following screen appears:



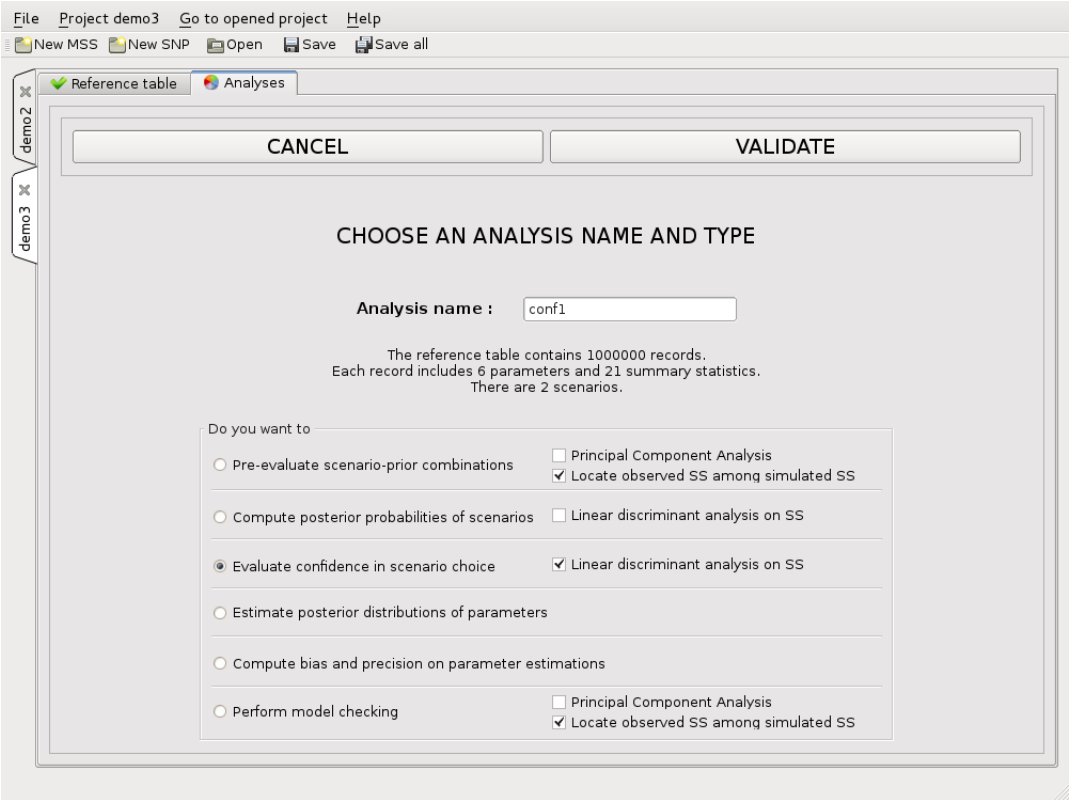
Both analyses agree that scenario 1 is the best supported scenario in this comparison. If we click on the View numerical results, the program shows a subset of numerical values used in the previous screen (i.e. the probability values with their 95% confidence intervals for the 10 subsets of closest simulated data). Note that the 95% CIs can be used to ensure that probabilities are significantly different among scenarios.



3.5.5 Confidence in scenario choice

This last type of analysis is aimed at evaluating with which level of confidence we can trust the previous analysis. To do so, we simulate test datasets (or pods), apply the same procedure for estimating their respective posterior probabilities and measure the proportion of times the right scenario has the highest posterior probability.

Let's define a new analysis, `conf1` as below:



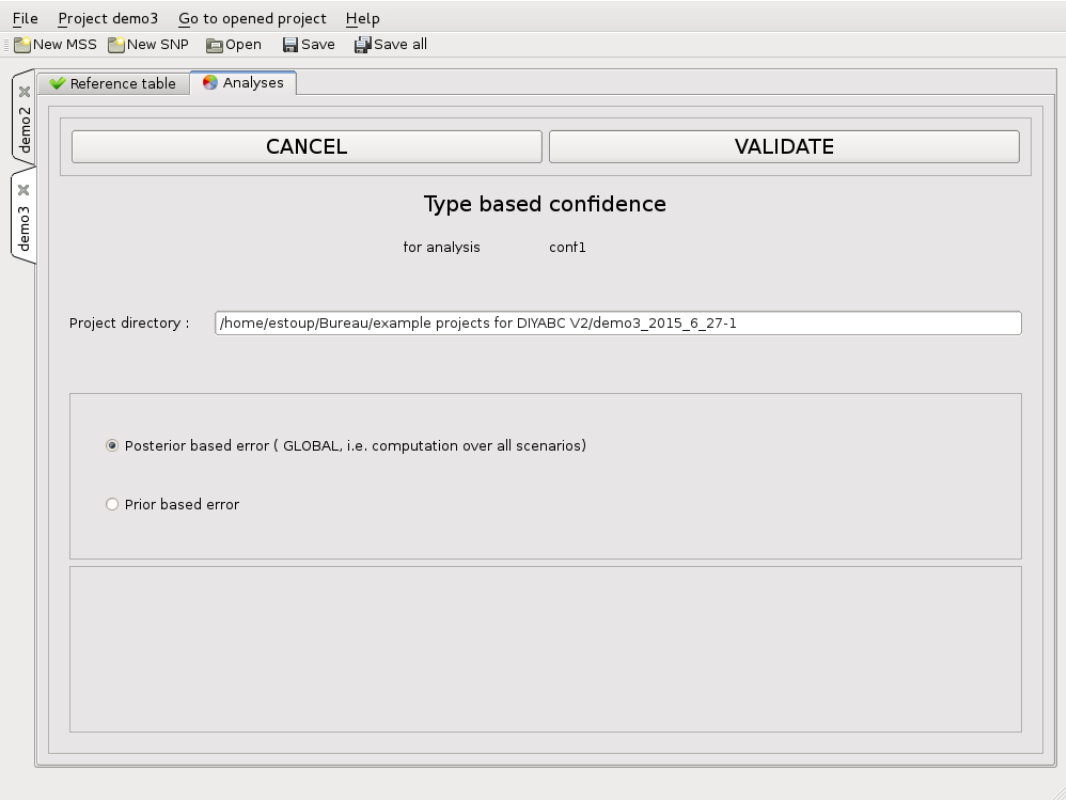
As for the previous analysis, it is possible to replace original summary statistics by discriminant scores (cf. "Linear discriminant analysis option" option; see Estoup et al. 2012). This is more useful here since confidence analyses can last hours. We hence choose to activate this option. If we do so, it is preferable to have previously computed the probabilities of scenarios from the observed dataset with the linear discriminant analysis option (cf. section 3.5.4) to homogenize treatments. Note that the computation of probabilities of scenarios from the observed dataset with the linear discriminant analysis option give similar results than those shown in section 3.5.4 (not shown).

The next screen (below) proposes two options :

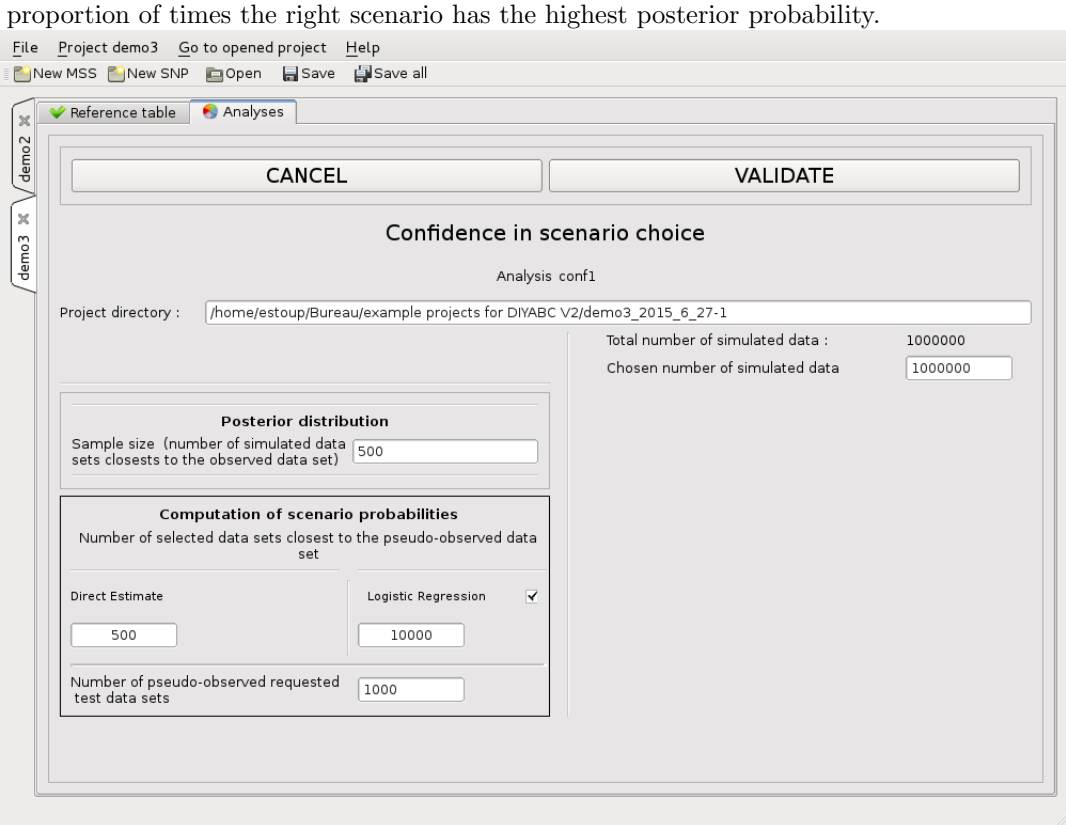
(i) Compute confidence in scenario choice drawing scenario-parameter combinations into posterior distributions (cf. Posterior based error);

(ii) Compute confidence in scenario choice drawing scenario-parameter combinations into prior distributions (cf. Prior based error).

Let's first consider *posterior based error computations*.



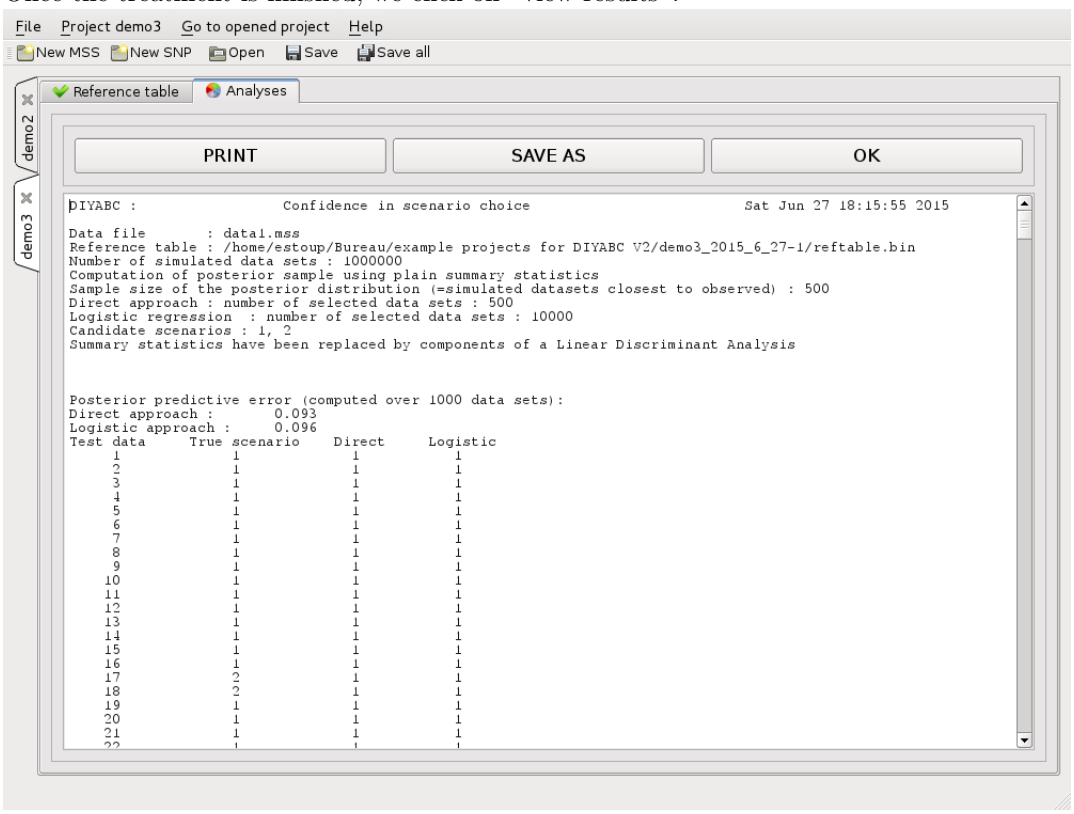
To compute "posterior" error rates, we simulate a large number of pseudo-observed datasets (pods) drawing (with replacement) the scenario ID and parameter values from the s simulated datasets closest to the observed dataset (i.e. the s datasets of the reference table with the smallest Euclidean distance). Typically, $s = 500$ but this number can be lowered to 100. For each pod produced this way, we apply the same procedure for estimating their respective posterior probabilities (as in section 3.5.4) and measure the proportion of times the right scenario has the highest posterior probability.



Here we choose to draw pods in the s=500 (over 1,000,000) simulated datasets closest to the observed dataset. Scenario probabilities are estimated using both the direct approach (on the 500 closest datasets)

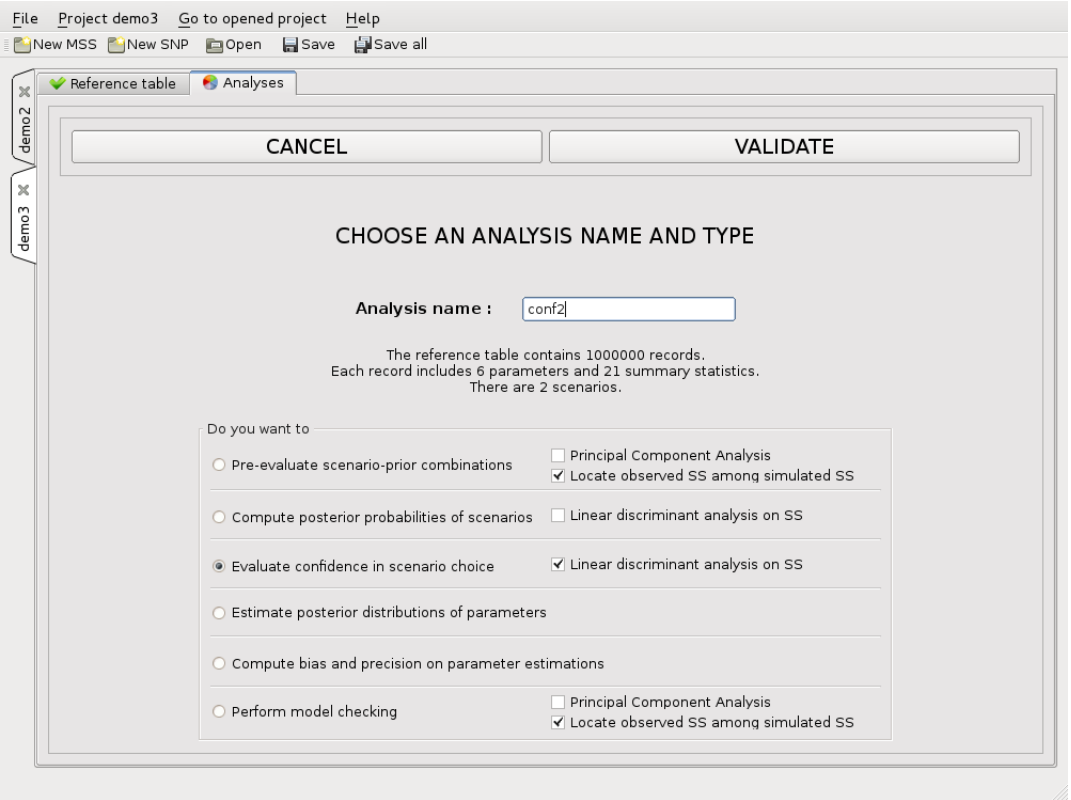
and the logistic approach (on the 1%=10,000 closest datasets). Computations are processed on a total of 1,000 test datasets (pods).

Once the treatment is finished, we click on "view results".



The *posterior error rate* (also named "posterior predictive error") is given as a proportion of wrongly identified scenarios over the 1,000 test datasets for both the direct and the logistic approaches. Here the true scenario had the highest posterior probability for 904 of the 1,000 test datasets with the logistic approach and the posterior error rate is hence equal to 0.096. The scenario choice is also detailed for each test dataset. Note the presence of a majority of scenario 1 and a minority of scenario 2 in the test datasets. This differencial proportion in scenario ID reflects the fact that a majority of scenario ID-parameters combination which give simulated datasets closest to the observed dataset are produced by the scenario 1 (which is expected when looking at the direct approach results in section 3.5.4). Computing error rate conditionally to the observed dataset (i.e. focusing around the observed dataset by using the posterior distributions) provide a more relevant estimation of our ability to choose the true scenario in the vicinity of the observed dataset (which is the location of prime interest in the vast data space defined by the prior distributions) than blindly computing accuracy indicator over the whole prior space.

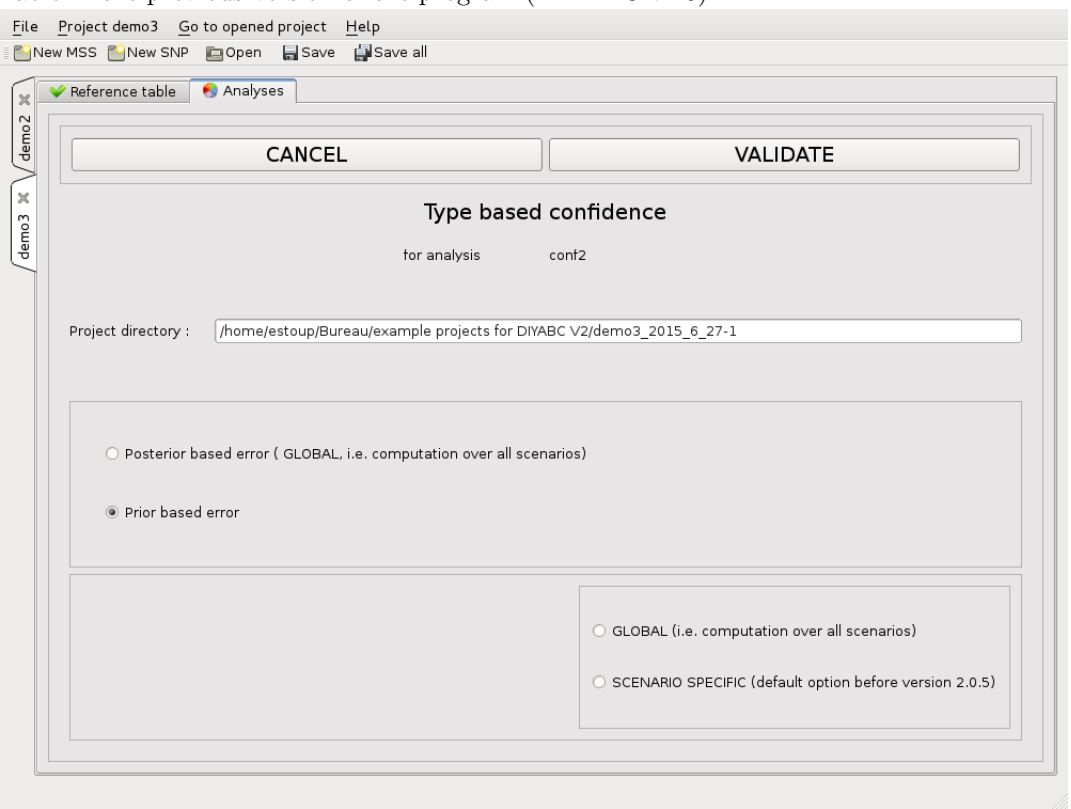
Let's now consider *prior based error computations*. Prior based error computation provides an estimate of a global error level over the whole (and usually huge) prior data space. Such computation can be useful for comparisons with the above posterior error rate, to focus investigation on a particular scenario and to select the best classifier and/or set of summary statistics (Pudlo et al. 2015). We start a new confidence analysis (`conf2`) from the analyses pannel below, using again the linear discriminant analysis option.



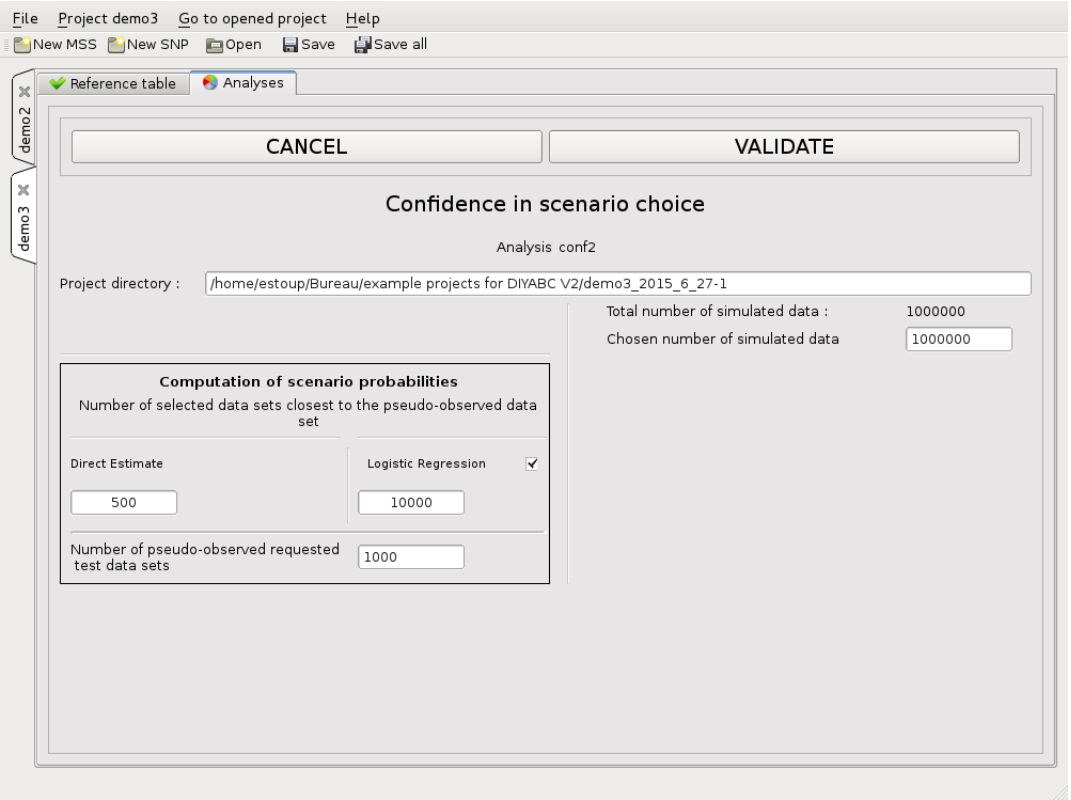
Once having clicked on "validate" and "Prior based error", two options are proposed:

(i) Global (prior error rate) in which pods are drawn from a random sample of scenario ID and parameter values in the prior distributions;

(ii) Scenario specific (prior error rate) in which pods are drawn from parameter prior distributions under a GIVEN scenario. This corresponds to the confidence in scenario choice option that was initially available in the previous version of the program (DIYABC v2.0).

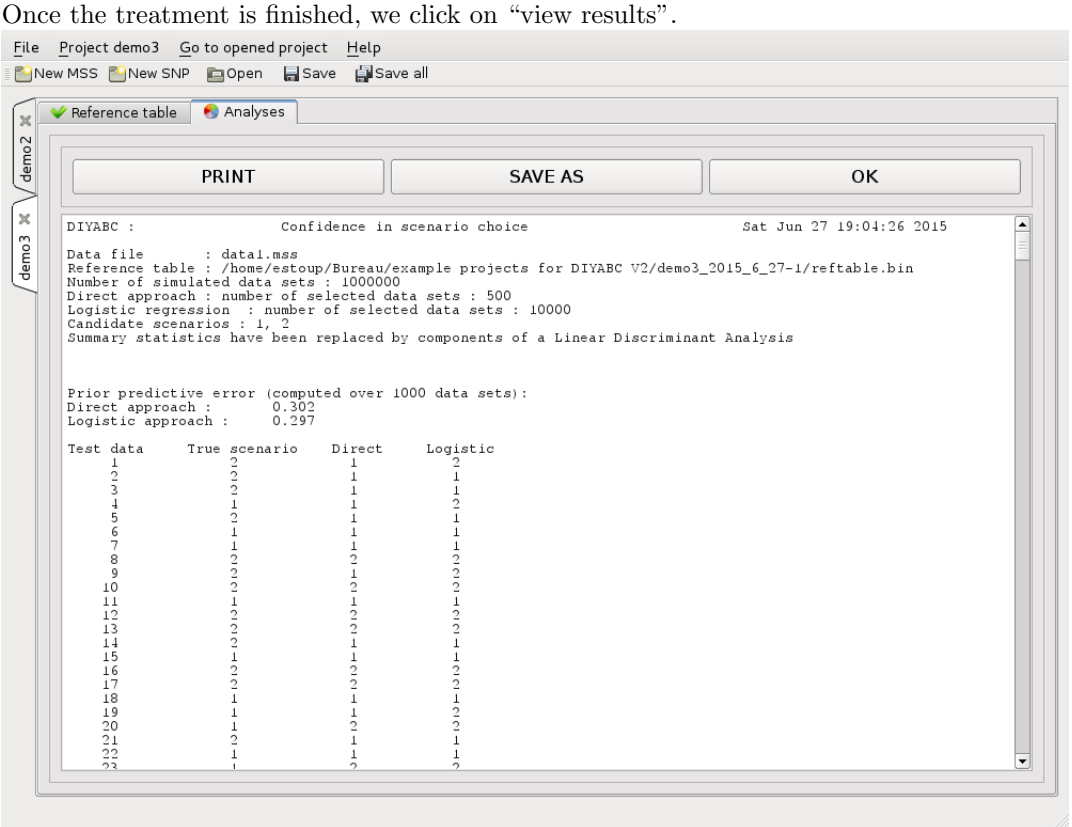


When clicking and validating the GLOBAL (prior error rate) option we go to the following screen.



This screen is similar to that for posterior error rate except that pods are NOT drawn from the s simulated datasets closest to the observed dataset BUT from a random sample of scenario ID and parameter values drawn in the prior distributions. We here again choose to estimate scenario probabilities using default options, i.e. using both the direct approach (on the 500 closest datasets) and the logistic approach (on the 1%=10,000 closest datasets), and computations are processed on a total of 1,000 test datasets (pods).

Once the treatment is finished, we click on "view results".

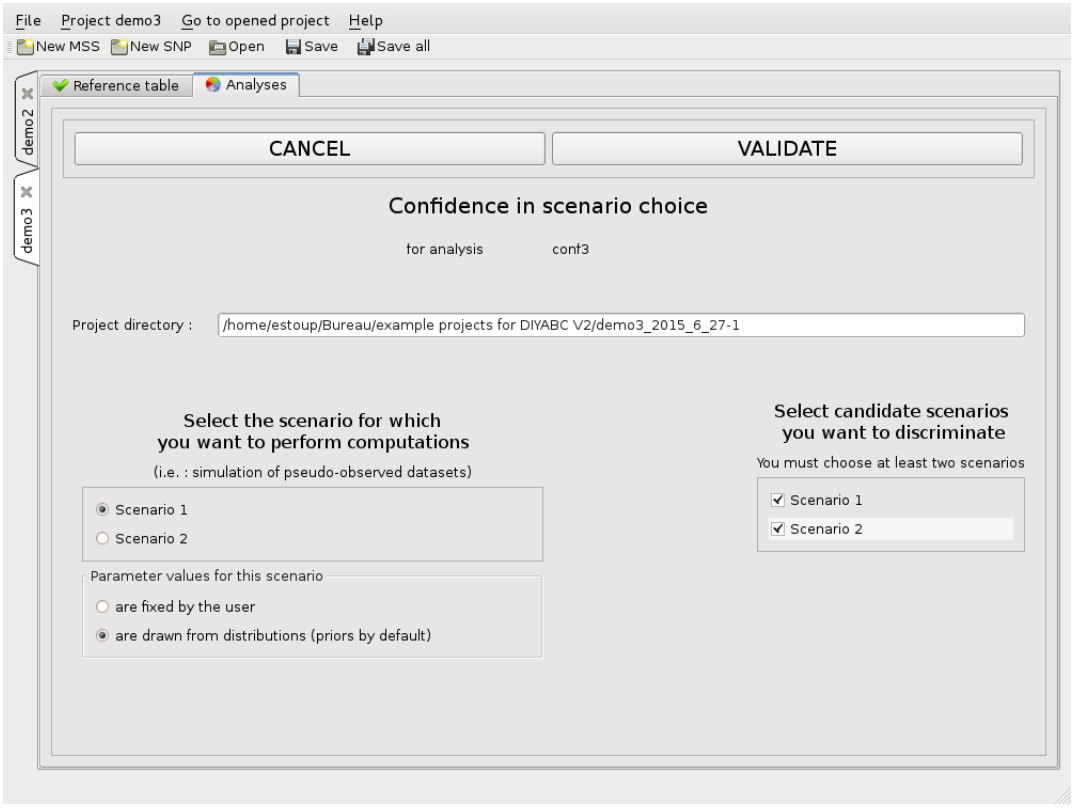


The prior error rate (also named "prior predictive error) is given as a proportion of wrongly identified

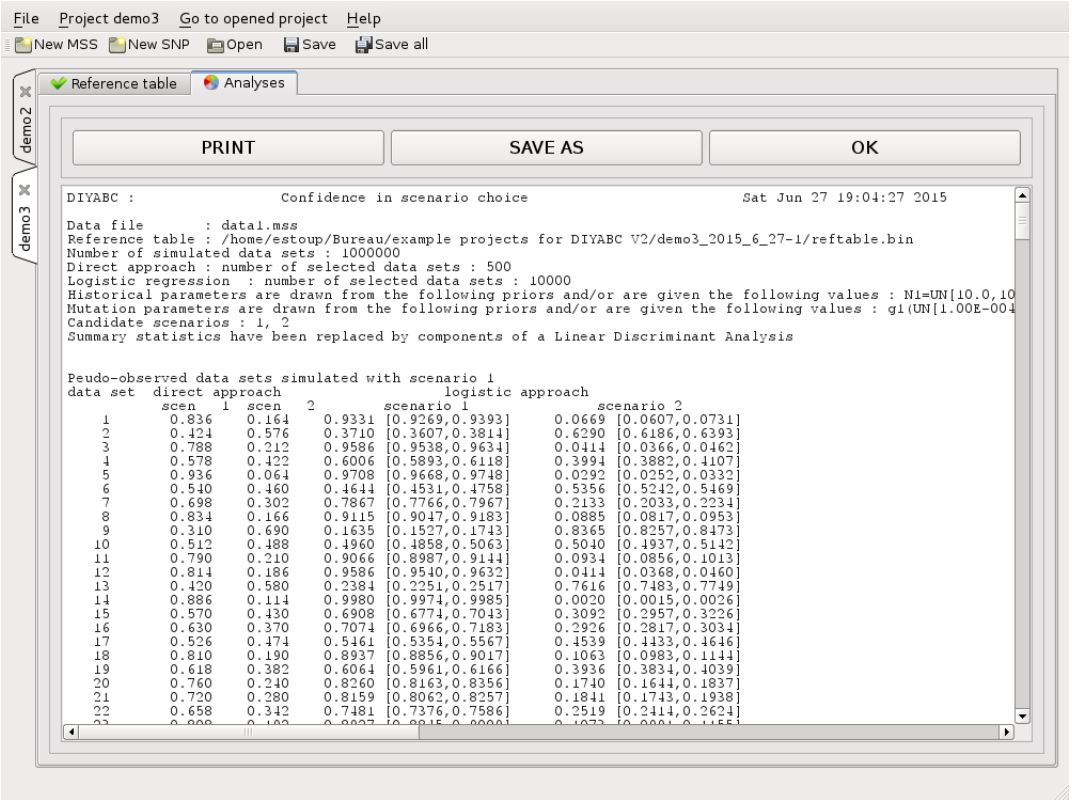
scenarios over the 1,000 test datasets for both the direct and the logistic approaches. The scenario choice is also detailed for each test dataset. Note the presence of a an equal number of scenario 1 and 2 in the test datasets as expected when randomly scenario ID and parameter values from the prior distributions. The prior error rate is substantially different than the previous posterior error rate (i.e. higher in this case although it might be lower in other situations). The error levels may indeed be substantially different depending on the location of the pod in the data space. Indeed, some peculiar combination of scenario ID and parameter values may correspond to situations of strong (weak) discrimination among the compared scenarios.

We now illustrate the second type of prior based error analysis: scenario specific (prior error rate) in which pods are drawn from parameter prior distribution under a GIVEN scenario. We start a new confidence analysis (`conf3`) and validate the options "Prior based error" + "Scenario specific (default option before version 2.1)".

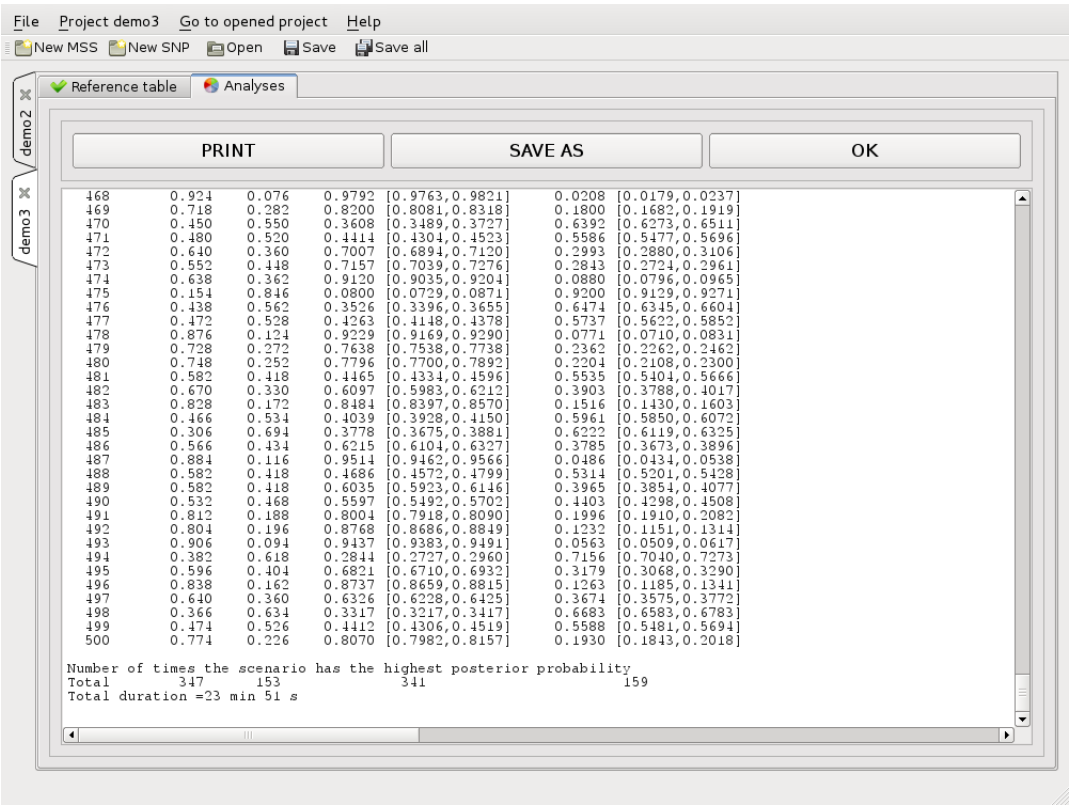
In the following screen we choose to simulate pods under scenario 1 drawing parameter values into prior distributions.



After validating default values for historical and mutational parameters, we launch the analysis. When it is done, we click on the View results button and get the following screen:



Posterior probabilities (with 95% credibility intervals) are given for each pod under the direct and logistic approaches. At the bottom, there is a summary of results, *i.e.* the number of times each scenario has the highest posterior probability under each approach:

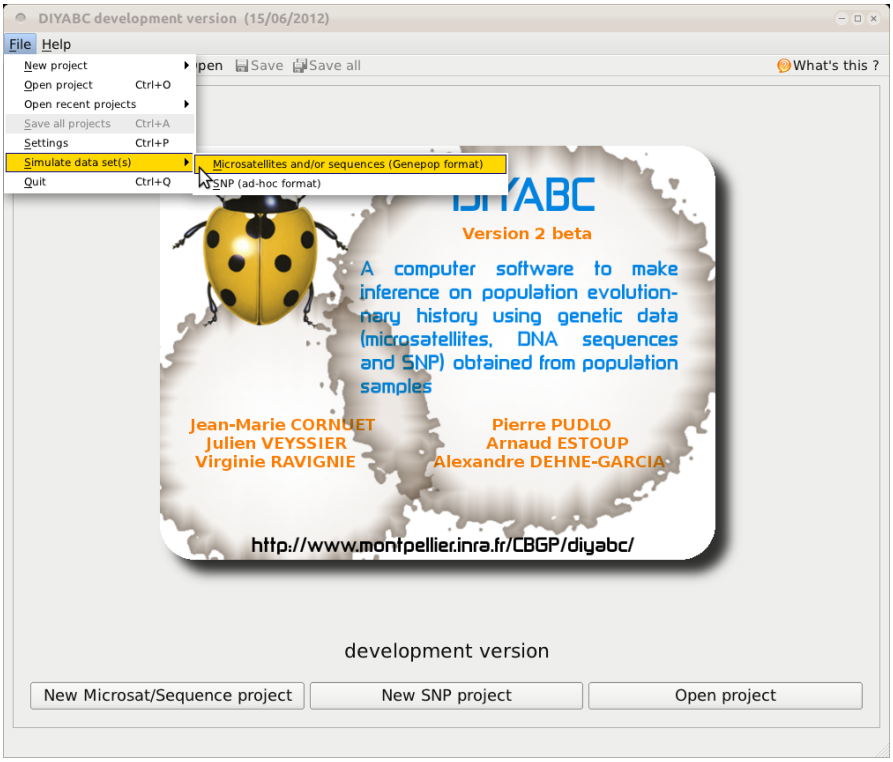


We can deduce the so-called *type I error for scenario 1*, which is the probability with which it is rejected although it is the true scenario : 153 using the direct method (or 159 using the regression

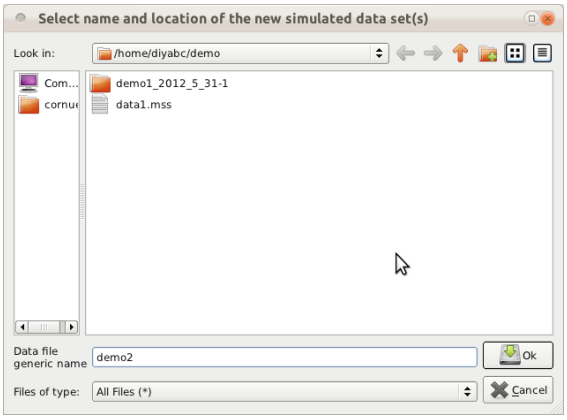
method) over 500, *i.e.* 0.306 (0.318). To have access to the type II error (probability of deciding for scenario 1 when it is not the true scenario), we need to run the same analysis but simulating according to all other scenarios (only scenario 2 in the present example) and counting decisions in favor of scenario 1. Running the example analysis with scenario 2 gives 165 (or 141) over 500 in favor of scenario 1. This gives an estimate of a so-called *type II error* of 0.330 (0.282) *for scenario 1*.

3.6 Simulating data sets

The DIYABC program can also be used to simulate data sets, either microsatellite and/or DNA sequence data sets using our Genepop format, or SNP data sets using our specific format. This option is reachable through the main File menu as shown below :

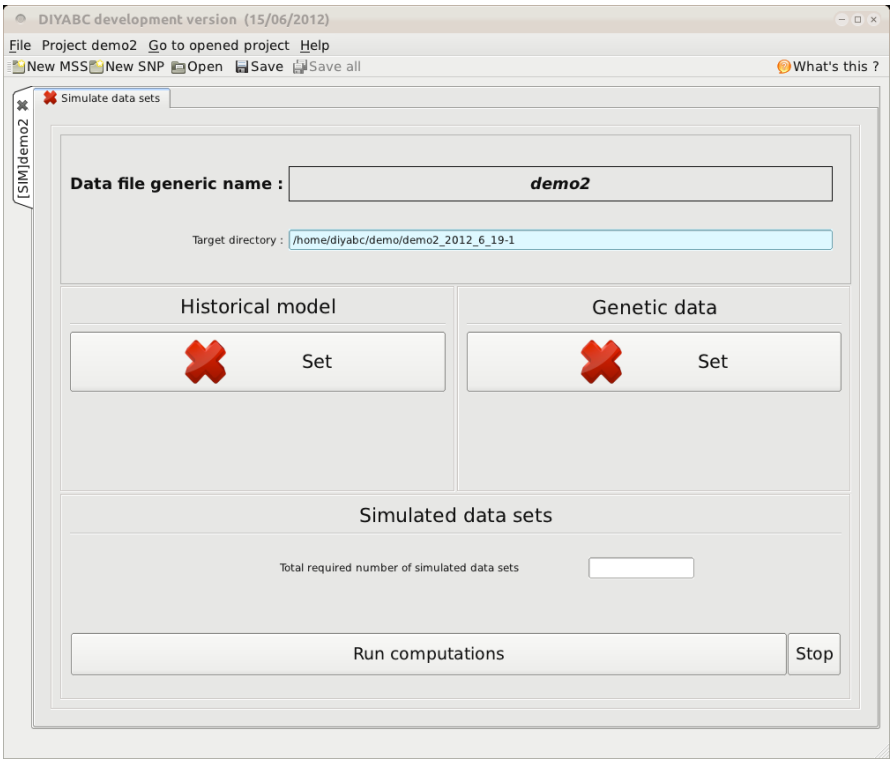


Clicking on e.g. the Microsatellites and/or sequences (Genepop format) opens up a dialog window in which one can choose the directory into which will be located the project and the future data files :

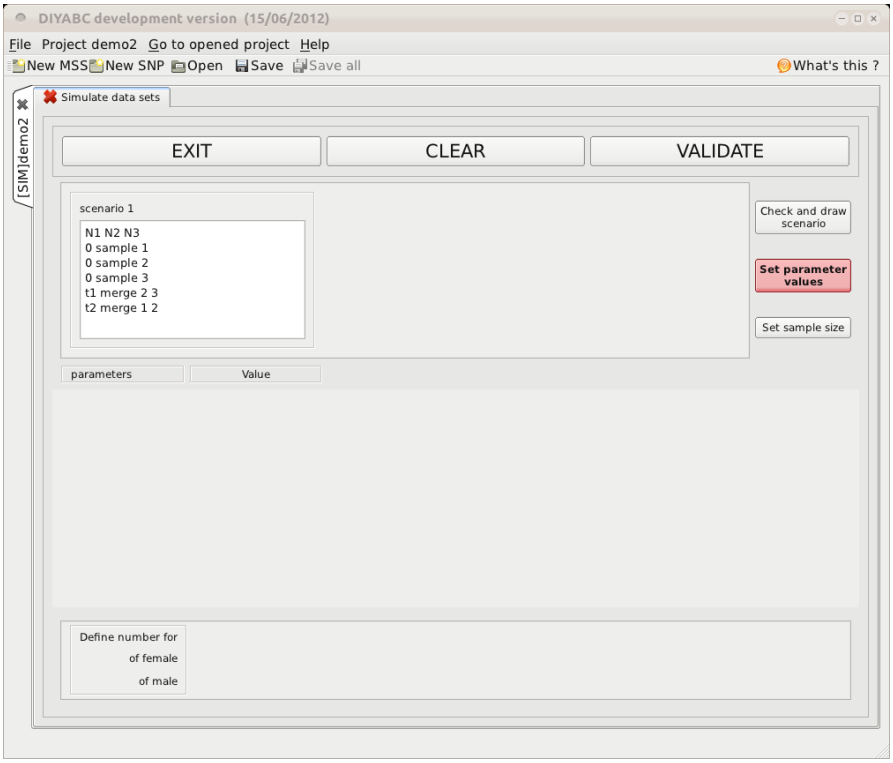


Above, we decided to call demo2 this new directory and to locate it in the home/DIYABC/demo directory.

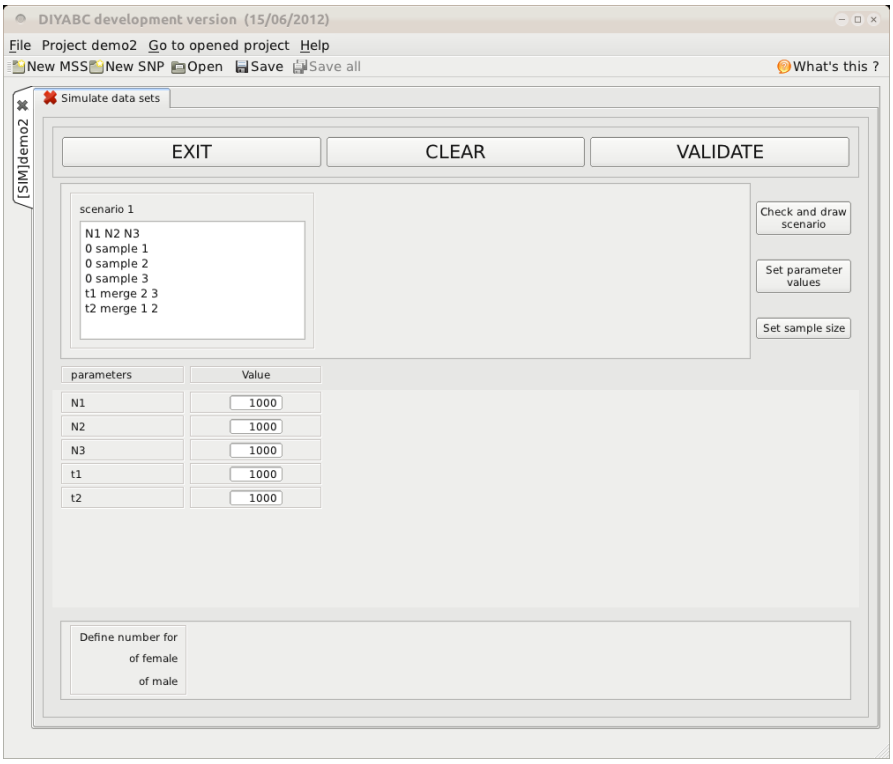
Clicking on OK leads to usual screen:



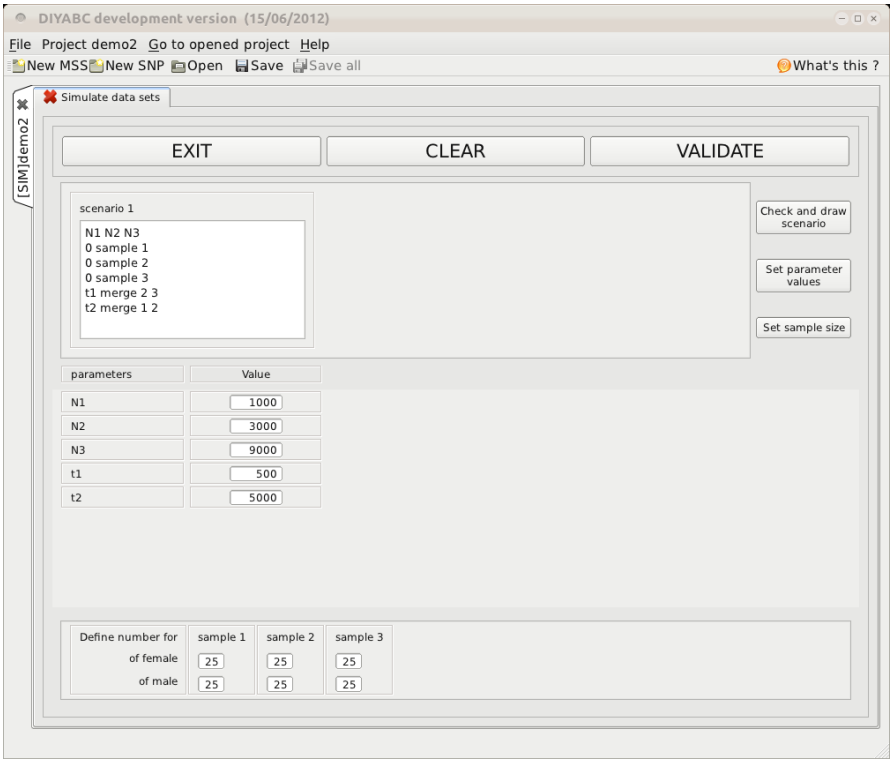
We first inform the historical model clicking on the Set button under Historical model. We edit the scenario box as below:



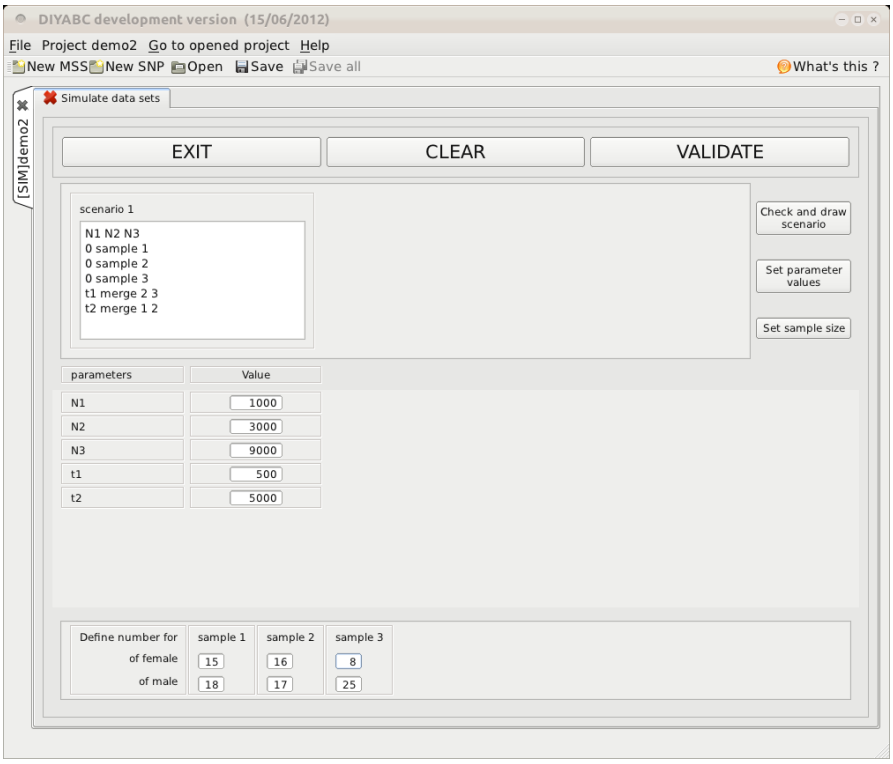
We click on the Set parameter values button. Arbitrary default values appear :



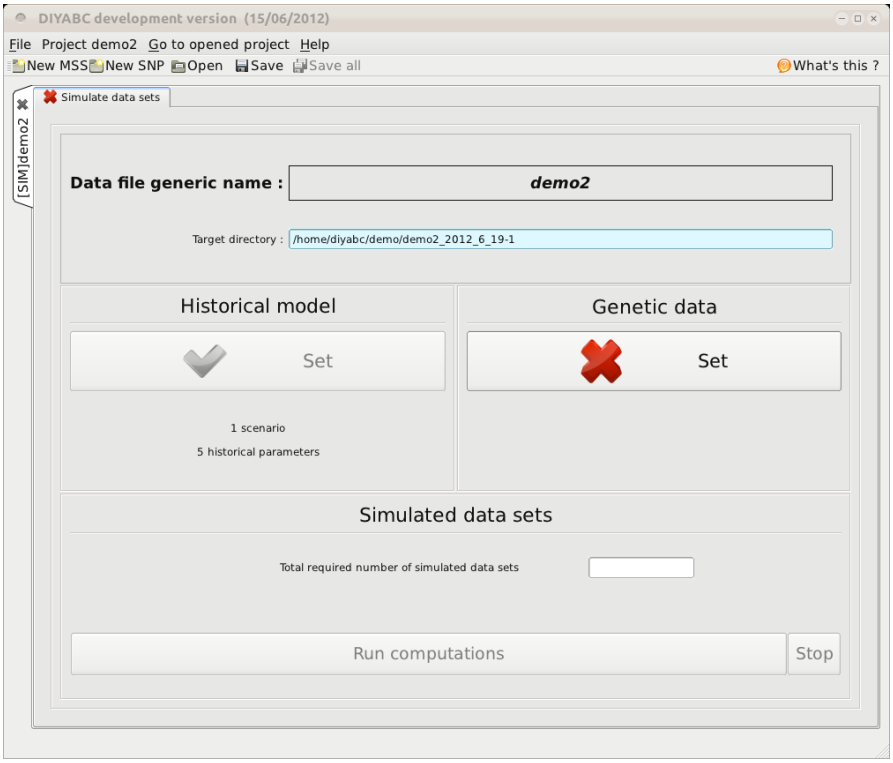
We change these values according to our needs and we click the Set sample size button, getting this screen:



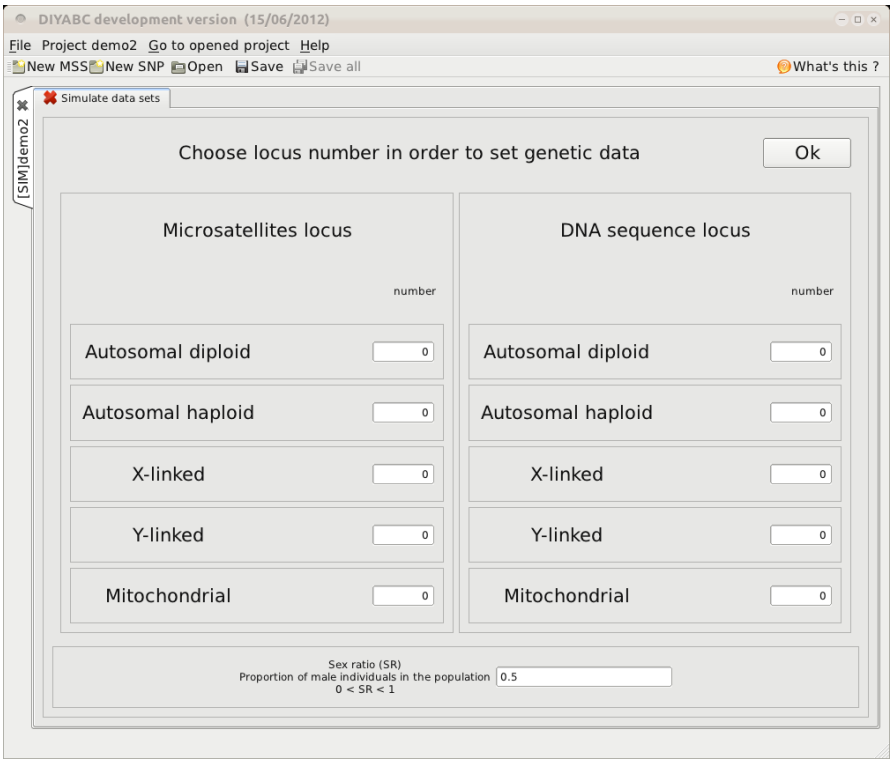
We input the needed sample sizes as below :



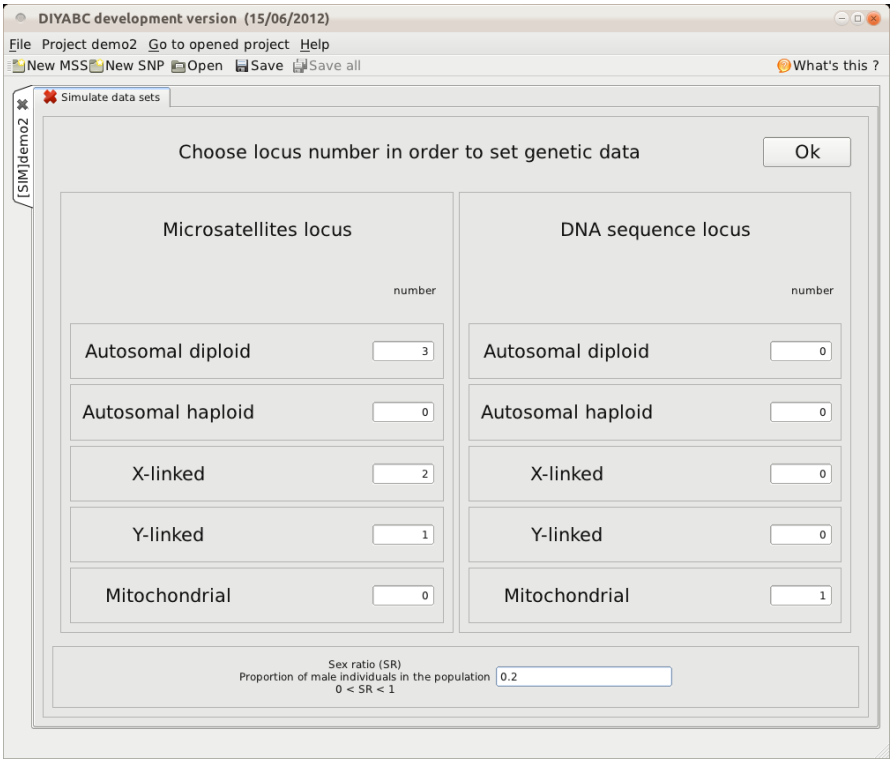
Clicking on the VALIDATE button, we get back to the previous screen showing that the Historical model is now completed:



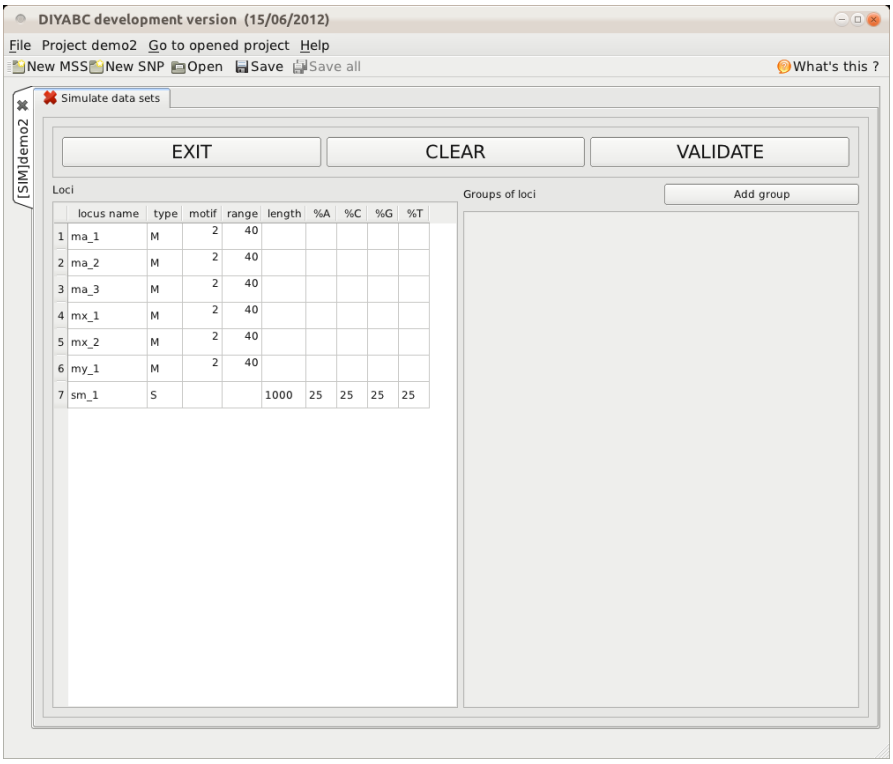
We have now to complete the Genetic data (click on the Set button under Genetic data). The following screen appears:



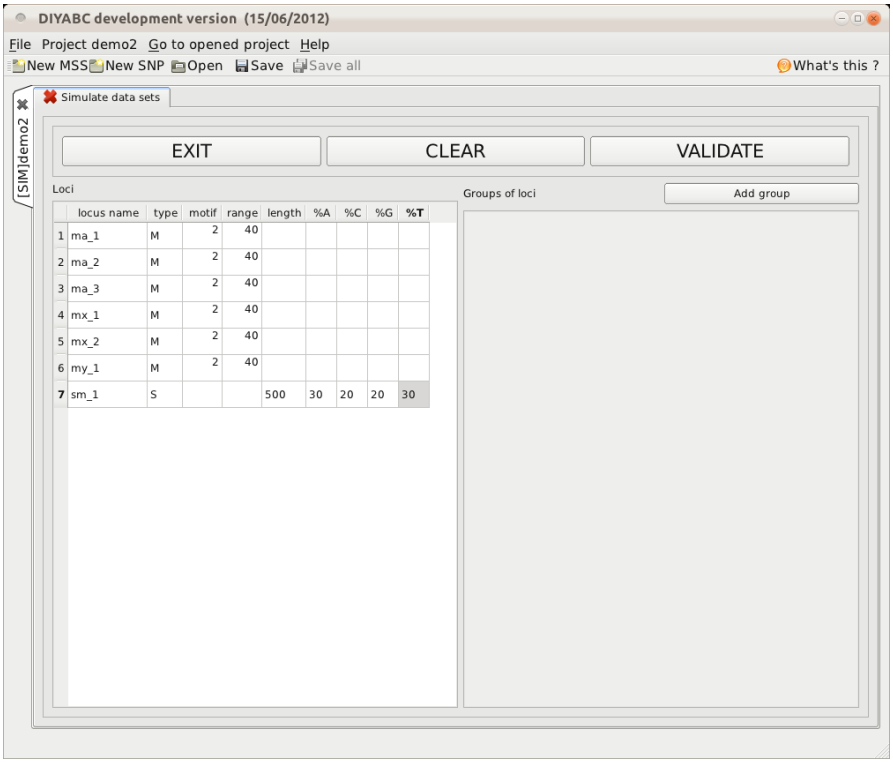
We want a data set including three autosomal, two X-linked and one Y-linked diploid microsatellite loci and one mitochondrial sequence. We also need a sex ratio of one male for four females :



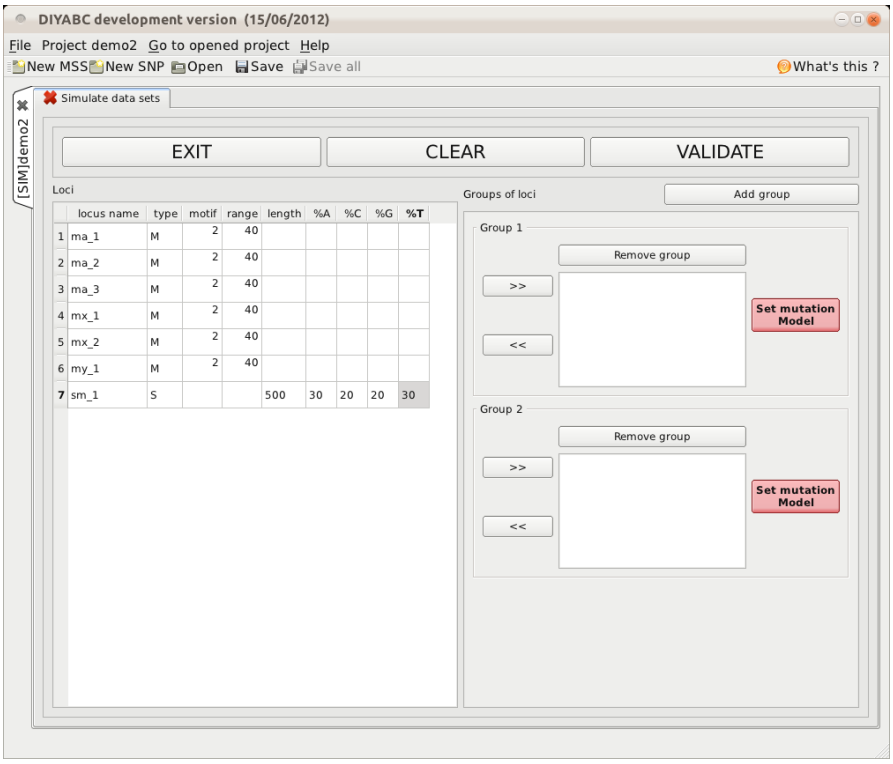
We click on the OK button and get the following screen :



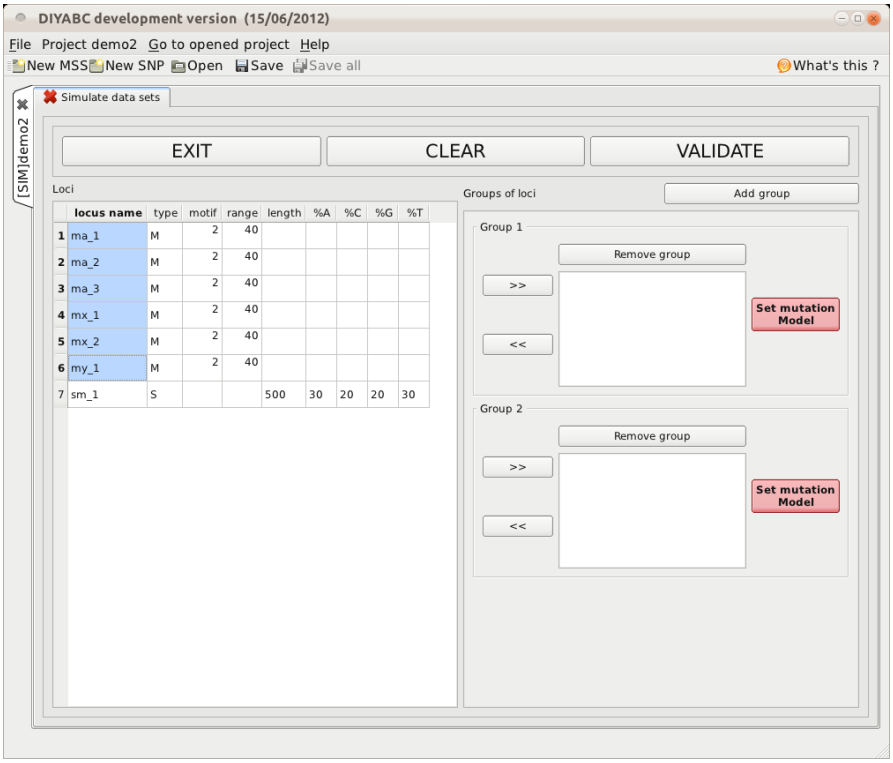
Our mitochondrial DNA sequence is only 500 nucleotides long and there is a slight excess of A+T (60%). We edit the corresponding cells :



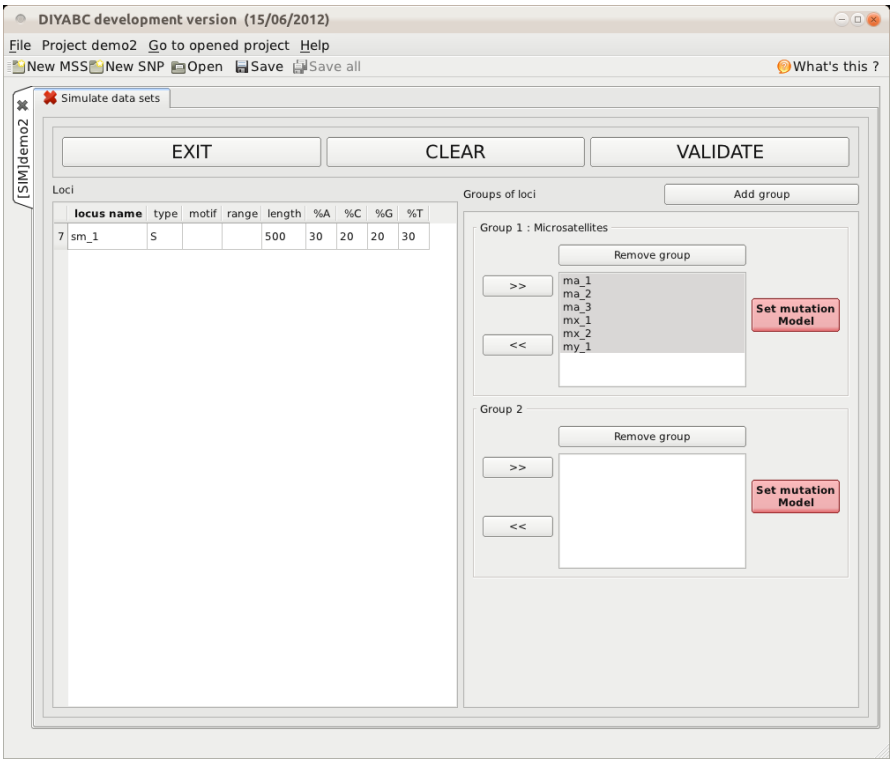
Since mutation models are different for microsatellites and DNA sequences, we define two groups by clicking twice on the Add group button :



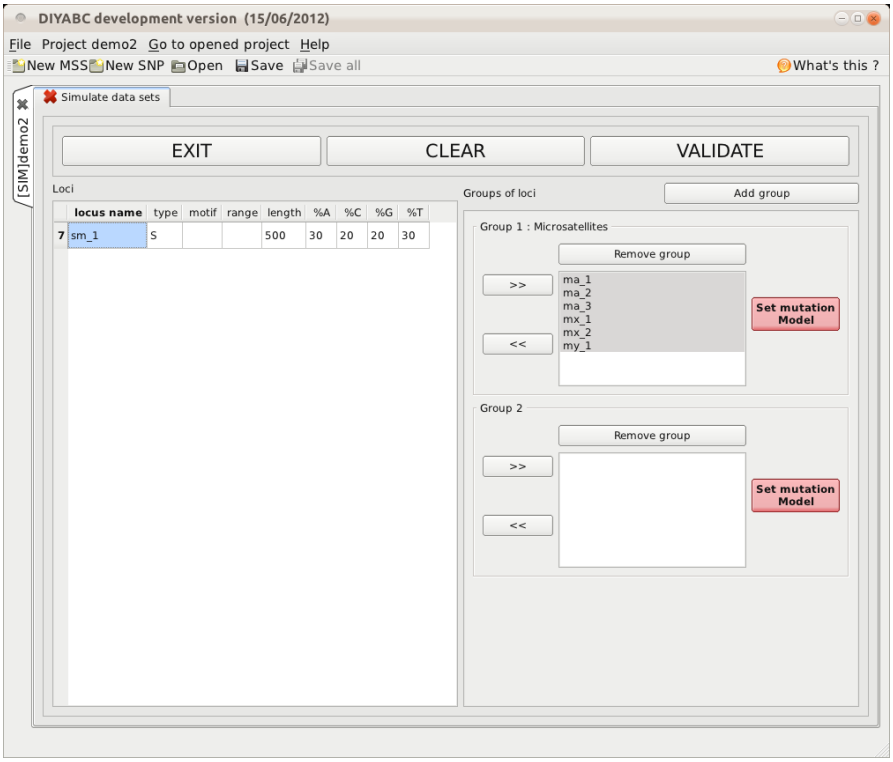
We select the 6 microsatellite loci by clicking on the first locus name cell and shift-clicking on the sixth locus name cell :



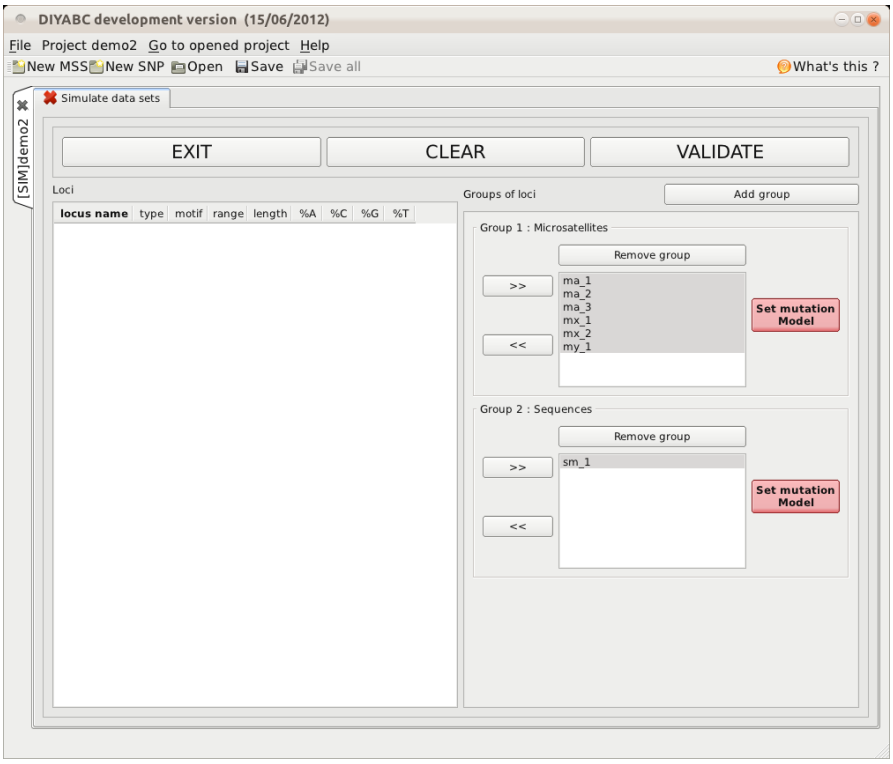
The six locus names are transferred into group 1 by clicking on the >> button :



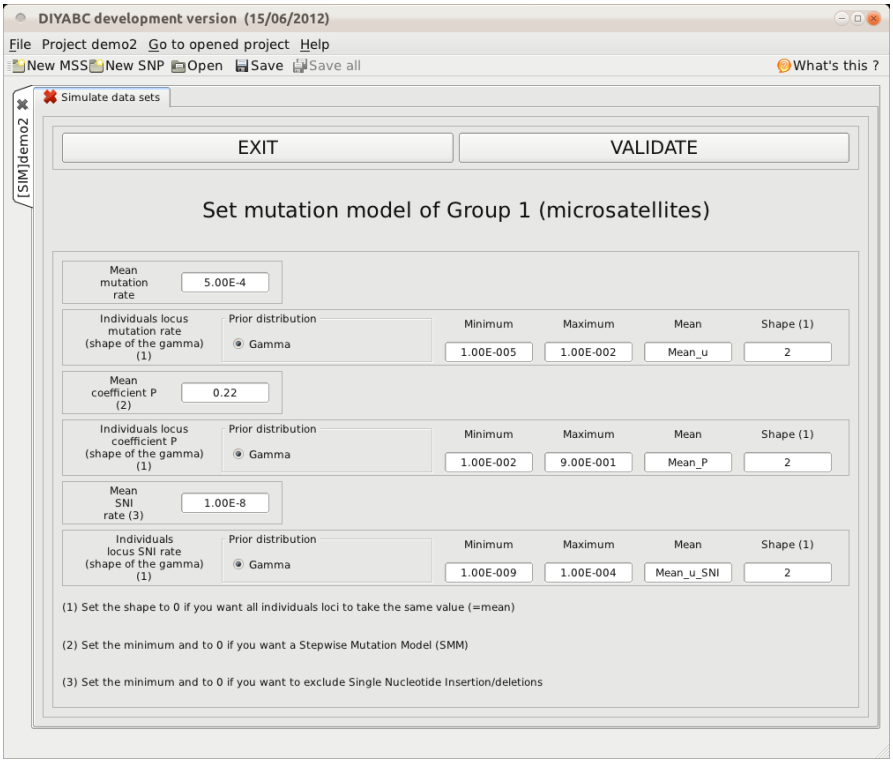
Then the DNA sequence locus is selected :



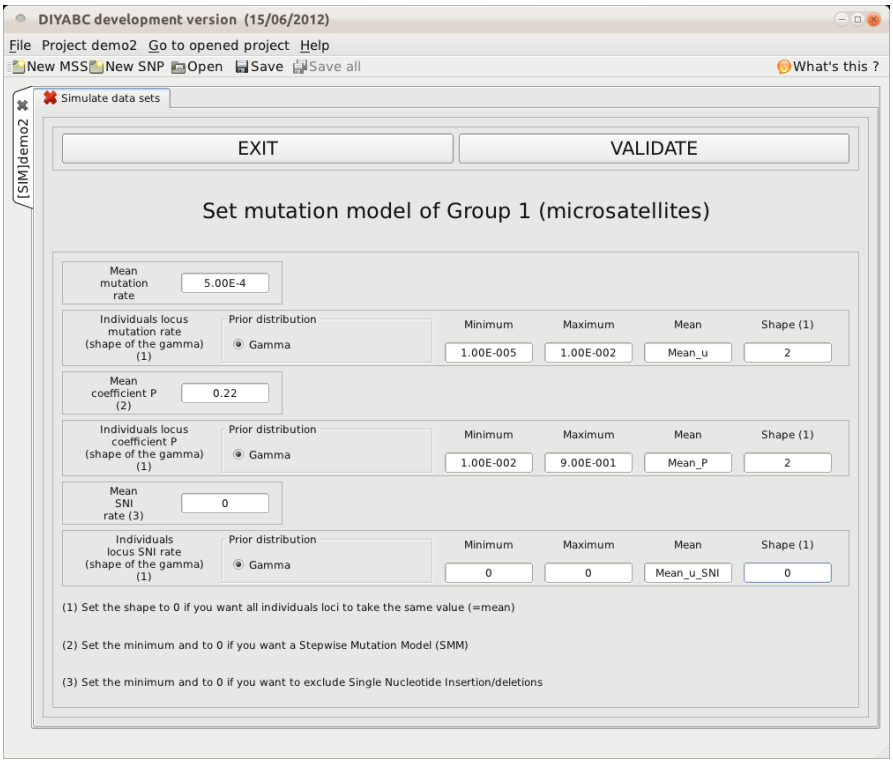
and transferred into group 2 in the same way :



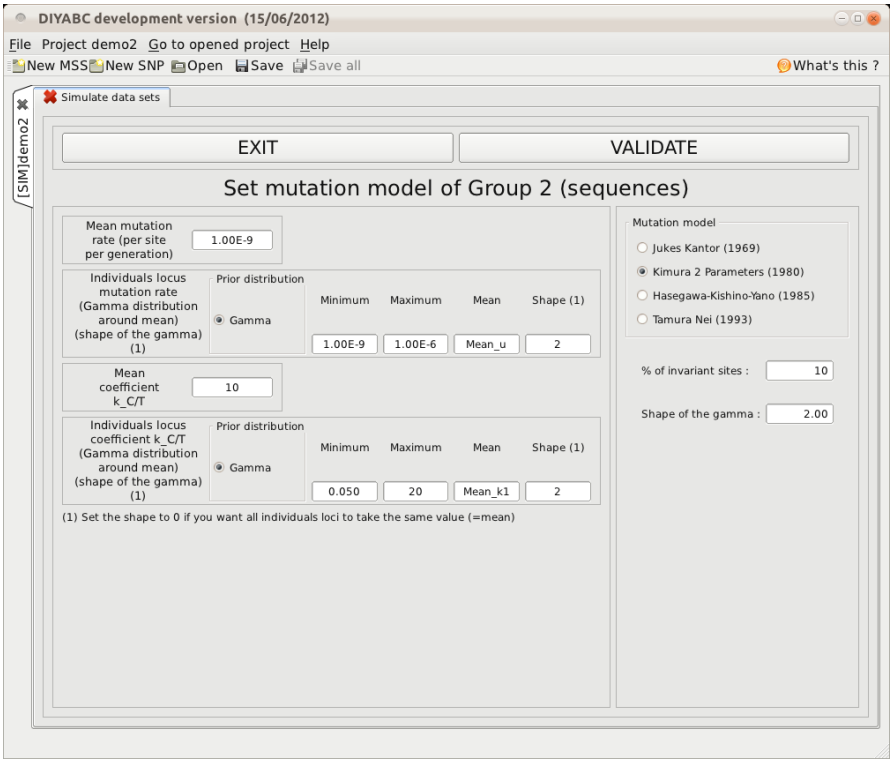
We need now to define the mutation model of each group (note that we not any mutation model needs to be defined for SNP loci cf. section 2.4). Let's click on the Set Mutation Model button of group 1:



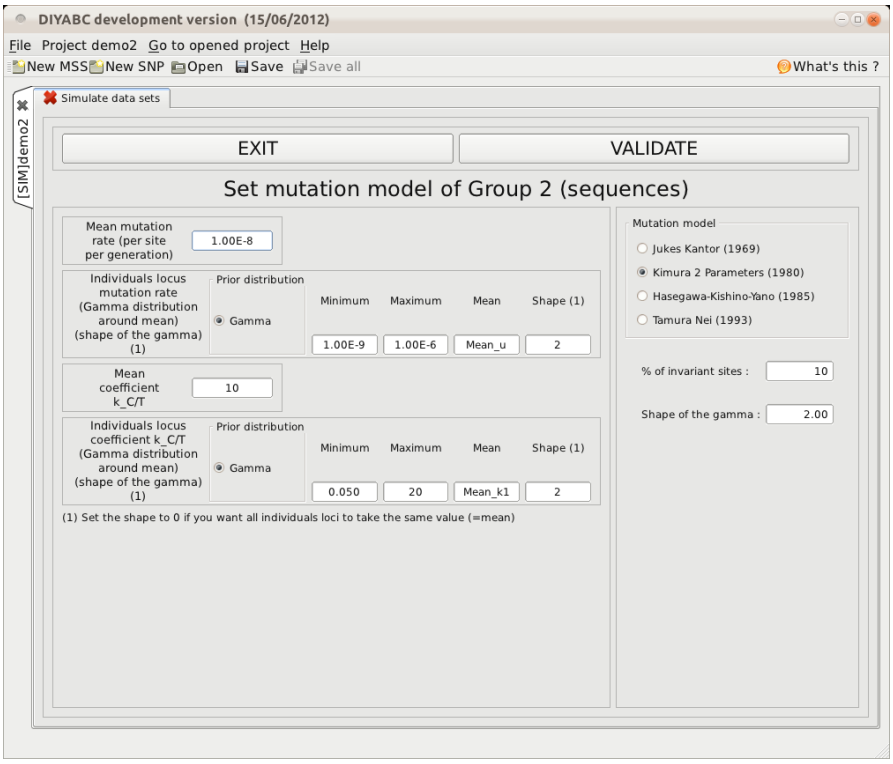
The usual default values appear. We want to exclude single nucleotide insertions/deletions (SNI mutations). So we set to 0 the Mean SNI rate and Minimum, Maximum and Shape of individual loci SNI rates :



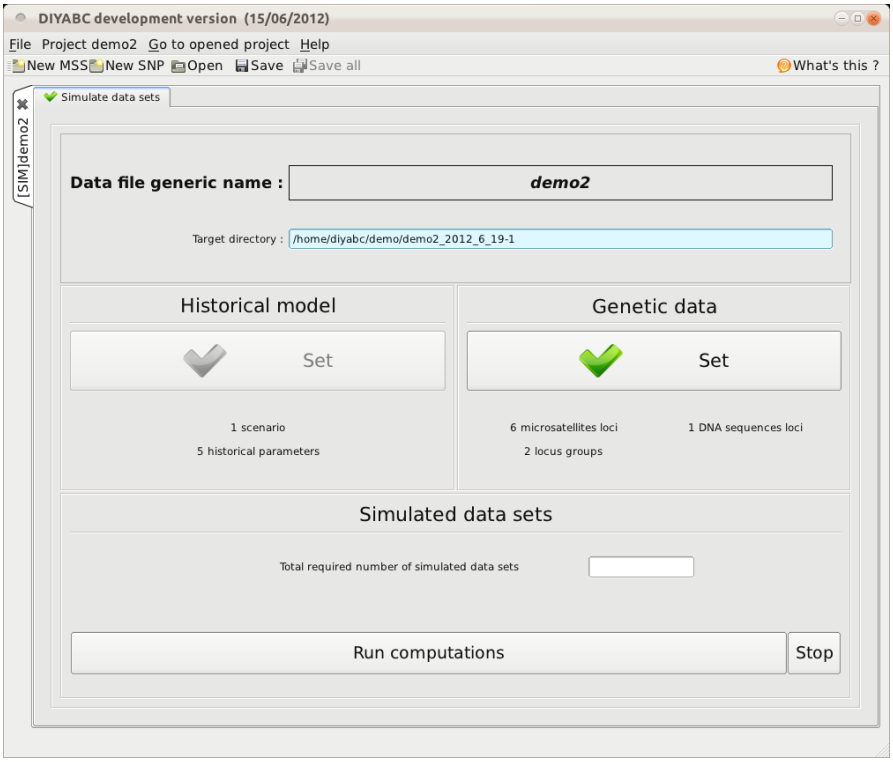
Once this done, we go back to the previous screen by clicking on the VALIDATE button. Then we set the mutation model of the mitochondrial DNA sequence. The default values are as follows :



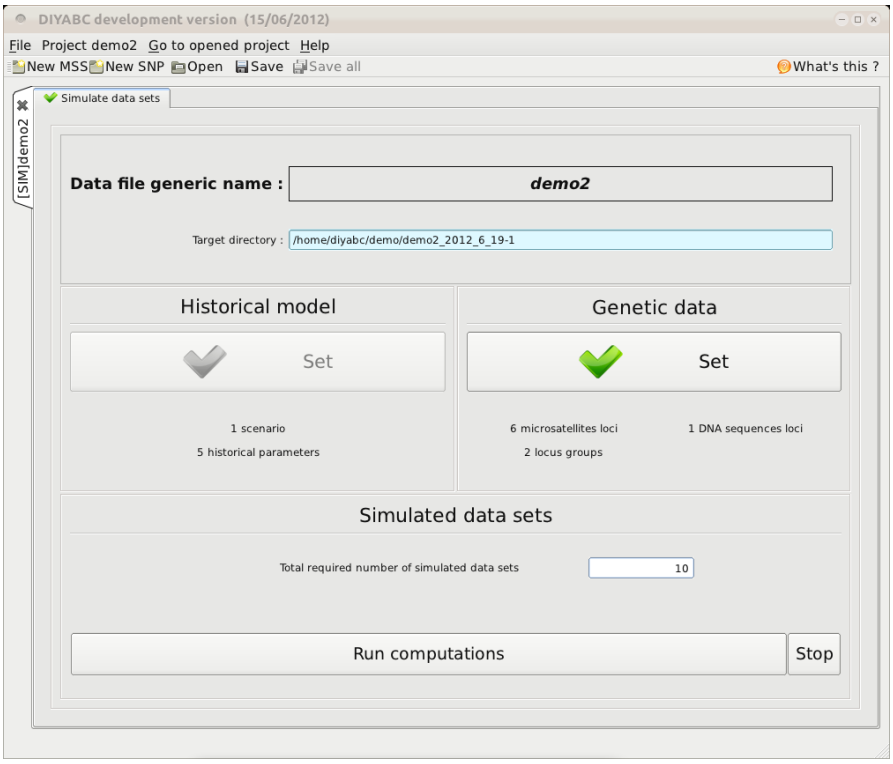
The default mean mutation rate is not suited to mitochondrial DNA which generally evolves at a faster rate than nuclear DNA (Haag-Liautard *et al.*, 2008). So we set its value to 10^{-8}. For all other parameters, we just keep the default values:



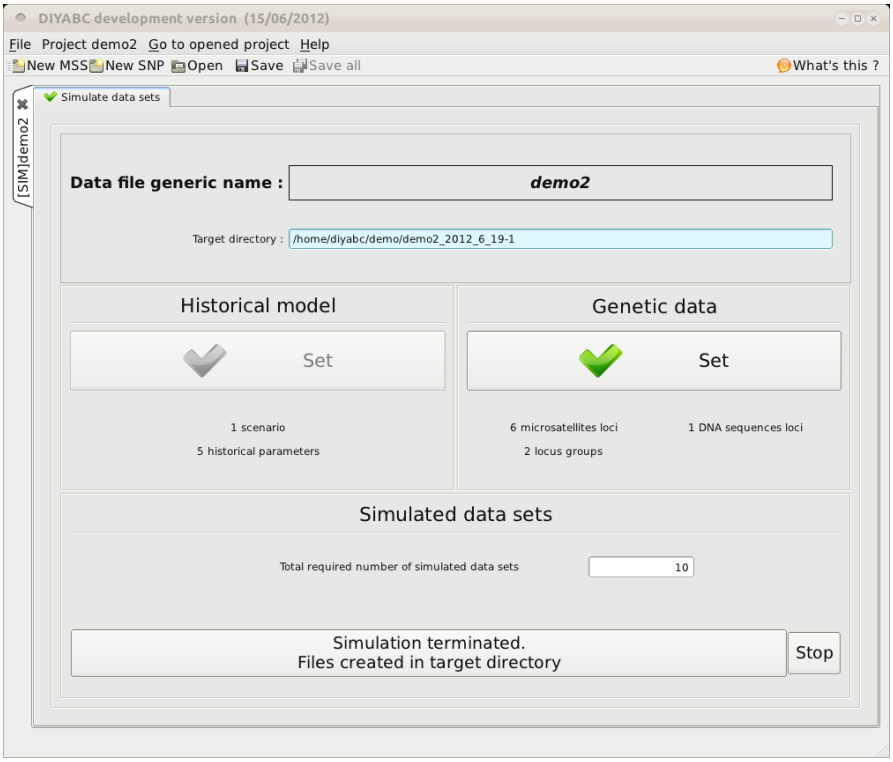
After validating twice, we get back to the main screen :



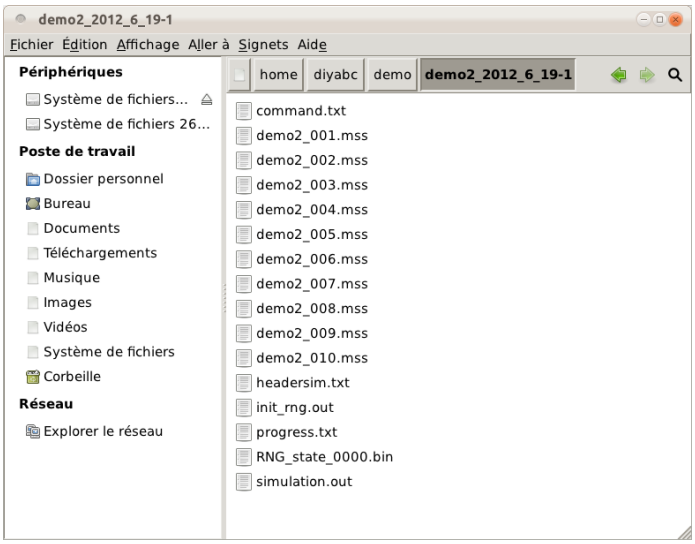
We require 10 simulated data sets :


DIYABC development version (15/06/2012)
File Project demo2 Go to opened project Help
New MSS New SNP Open Save Save all
What's this ?
[SIM]demo2
Simulate data sets
Data file generic name :
demo2
Target directory :
/home/diyabc/demo/demo2_2012_6_19-1
Historical model
Set
1 scenario
5 historical parameters
Genetic data
Set
6 microsatellites loci
2 locus groups
1 DNA sequences loci
Simulated data sets
Total required number of simulated data sets
10
Run computations
Stop

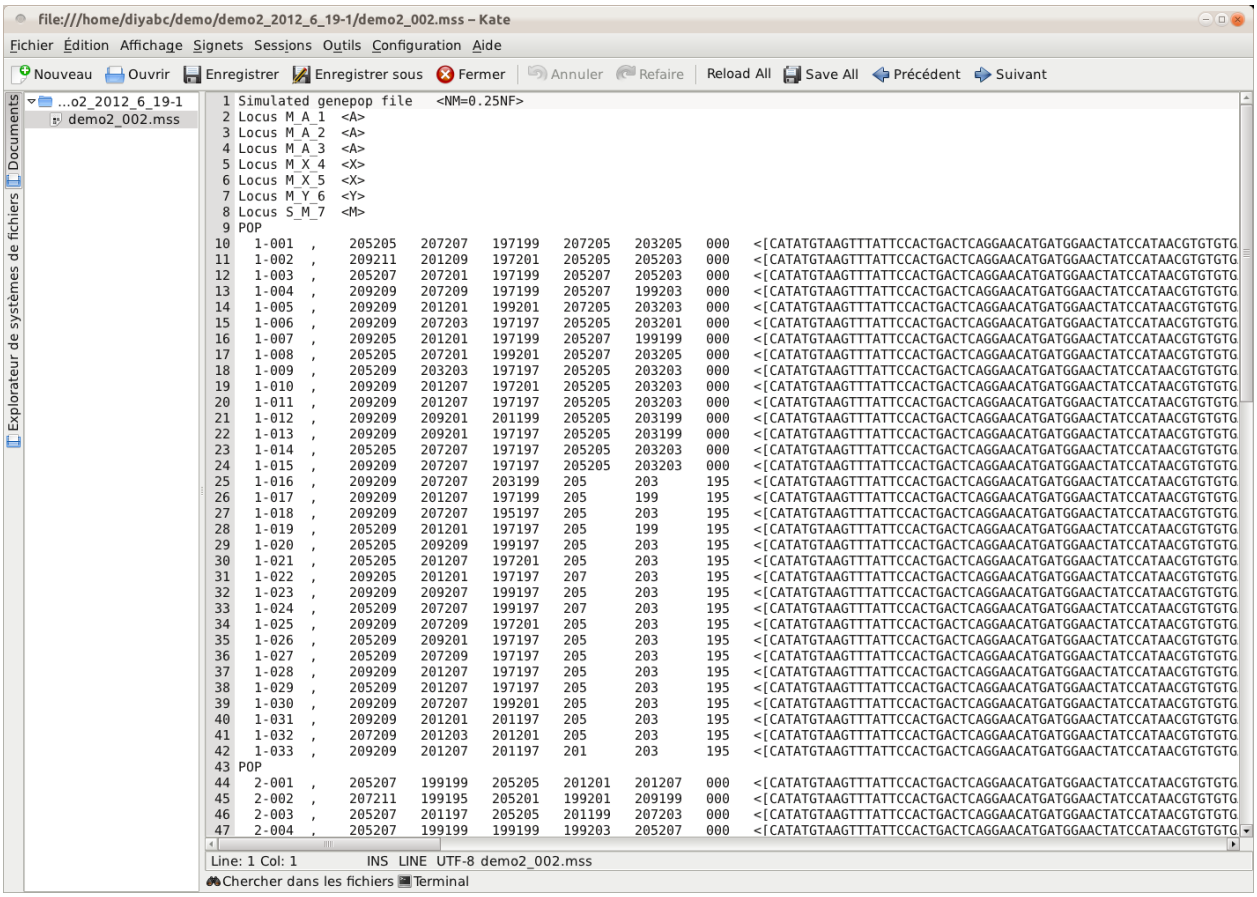
We then click on the Run computation button. In a matter of seconds, the computation ends up:



Using the file manager, we can check that ten new files (demo2_001.mss to demo2_010.mss) have been added to new directory :



Opening e.g. the second one with a text editor, we can have a partial view of the simulated genotypes of the first population sample :



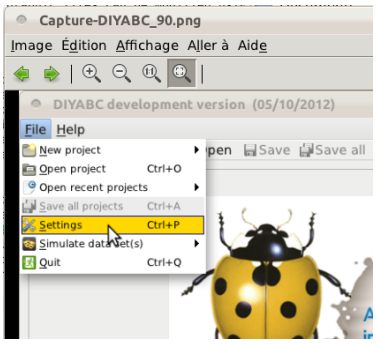

```
Simulated genepop file   <NM=0.25NF>
Locus M_A_1  <A>
Locus M_A_2  <A>
Locus M_A_3  <A>
Locus M_X_4  <X>
Locus M_X_5  <X>
Locus M_Y_6  <Y>
Locus S_M_7  <M>
POP
  1-001  ,    205205   207207   197199   207205   203205   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-002  ,    209211   201209   197201   205205   205203   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-003  ,    205207   207201   197199   205207   205203   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-004  ,    209209   207209   197199   205207   199203   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-005  ,    209209   201201   199201   207205   203203   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-006  ,    209209   207203   197197   205205   203201   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-007  ,    209205   201201   197199   205207   199199   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-008  ,    205205   207201   199201   205207   203205   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-009  ,    205209   203203   197197   205205   203203   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-010  ,    209209   201207   197201   205205   203203   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-011  ,    209209   201207   197197   205205   203203   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-012  ,    209209   209201   201199   205205   203199   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-013  ,    209209   209201   197197   205205   203199   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-014  ,    205205   207207   197197   205205   203203   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-015  ,    209209   207207   197197   205205   203203   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-016  ,    209209   207207   203199   205      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-017  ,    209209   201207   197199   205      199      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-018  ,    209209   207207   195197   205      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-019  ,    205209   201201   197197   205      199      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-020  ,    205205   209209   199197   205      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-021  ,    205205   201207   197201   205      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-022  ,    209205   201201   197197   207      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-023  ,    209209   209207   199197   205      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-024  ,    205209   207207   199197   207      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-025  ,    209209   207209   197201   205      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-026  ,    205209   209201   197197   205      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-027  ,    205209   207209   197197   205      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-028  ,    209209   201207   197197   205      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-029  ,    205209   201207   197197   205      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-030  ,    209209   207207   199201   205      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-031  ,    209209   201201   201197   205      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-032  ,    207209   201203   201201   205      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  1-033  ,    209209   201207   201197   201      203      195   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
POP
  2-001  ,    205207   199199   205205   201201   201207   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  2-002  ,    207211   199195   205201   199201   209199   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  2-003  ,    205207   201197   205205   201199   207203   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
  2-004  ,    205207   199199   199199   199203   205207   000   <[CATATGTAAGTTTATTCCACTGACTCAGGAACATGATGGAACTATCCATAACGTGTGTG
```

We can check that the sex ratio is correct : the number of males is one fourth the number of females. The type of each locus given after the name is also correct. All microsatellite allelic values are odd, in agreement with the motif length (2) and the absence of single nucleotide insertion/deletion. More interestingly, it gives an example of how X- and Y-linked microsatellite loci must be written for each sex (here 15 females and 18 males) in our Genepop format.

In the same spirit and following similar implementation steps, the option "Simulate dataset(s)" allows producing SNP dataset files too (see the first demonstration screen of this section 3.6). The SNP data are produced following the Hudson's simulation algorithm (Hudson 2002; Cornuet et al. 2014). Each locus will hence be characterized by the presence of at least a single copy of a variant over all genes sampled from all studied populations (i.e. pooling all genes genotyped at the locus). The format of the produced SNP genotype datasets is the DIYABC format chosen for SNPs and detailed in section 4.4. The produced dataset(s) can hence be directly analyzed using DIYABC as pseudo-observed dataset(s) for which the scenario and the parameter values are known. Note that it is possible to subsequently apply a different MAF criterion on the pseudo-observed dataset before running an ABC analysis by replacing in the headline of the pseudo-observed dataset the instruction <MAF=hudson> by <MAF=*X.XX*> (for instance <MAF=0.05>).

3.7 The **Settings** option of the **File** menu

Let us now detail what is under the Settings option of the File menu shown below :


Capture-DIYABC_90.png
Image Édition Affichage Aller à Aide
DIYABC development version (05/10/2012)
File Help
New project
Open project Ctrl+O
Open recent projects
Save all projects Ctrl+A
Settings Ctrl+P
Simulate dataset(s)
Quit Ctrl+Q
Save
Save all

Clicking on the Settings option opens up the following multitab window :

3.7.1 Tab "various"

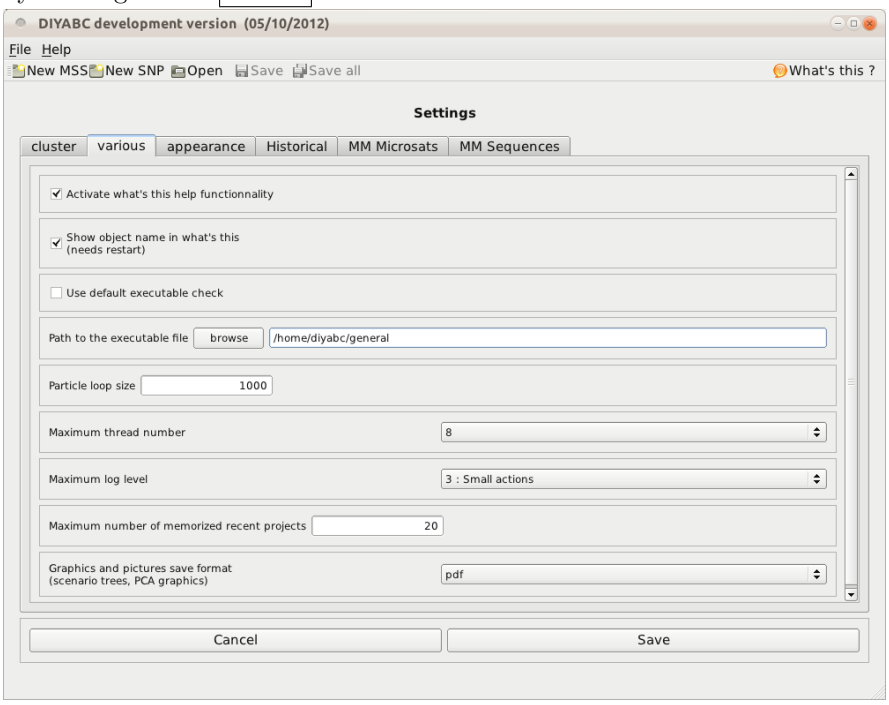
The first tab "various" contains the following settings :

1. What's this is a help functionnality that allows the user to obtain a help message when pointing towards a specific feature of the graphic interface such as a button or an edit field. This help functionnality can be activated by checking the corresponding box.

2. Checking this box is mainly for debugging purpose or signalling a bug.

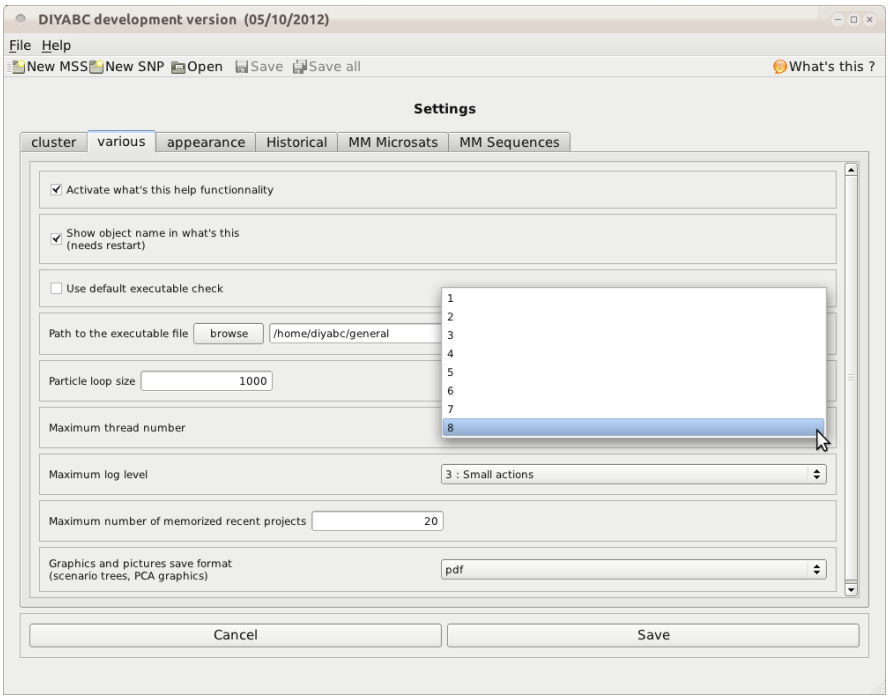
3. DIYABC is made of two programs : the graphic interface and a computation program. When the user clicks on buttons such as Run computation or Launch, the graphic interface programs sends a command that launches the computation program. To issue this command, the graphic interface needs to know where the computation program executable is located. There is a default location which depends on the operating system. Clicking on the box Use default executable check will direct the graphic interface to use the executable located in this default directory.

4. You can also choose another location (e.g. if you want to use a distinct version of the executable) by clicking on the browse button.



5. The next setting Particle loop size defines the number of data sets (n) that are simulated in a single block when building the reference table. The computation program proceeds as follows : it first simulate and compute summary statistics of n data sets. When this is done, it writes the results to the reference table file. The reason of doing like this is that computation can be multithreaded but not the file writing.

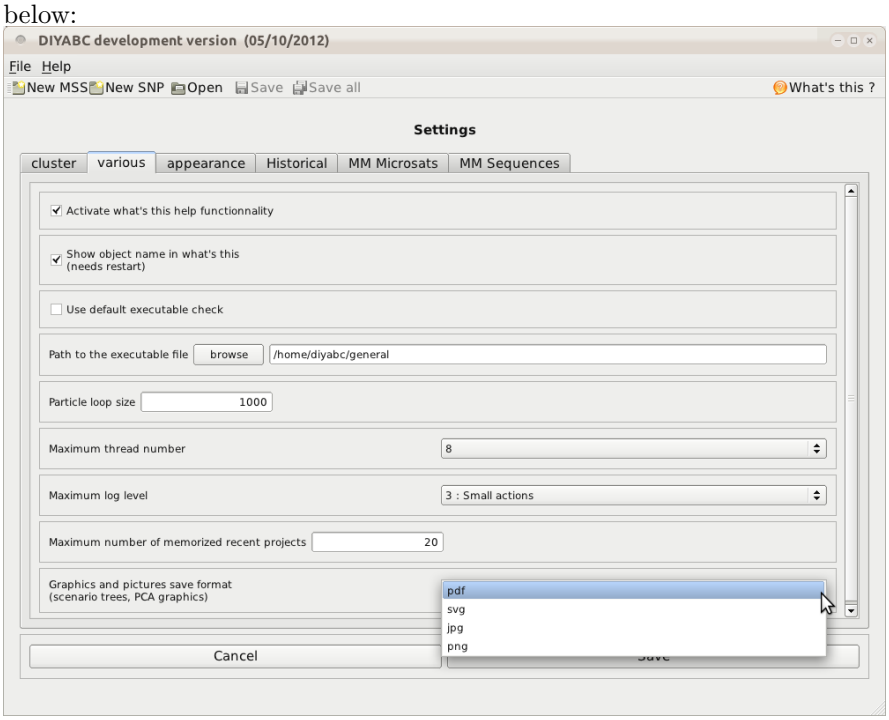
6. The graphic interface can detect the number of cores of the computer processor. By default, it sets the number of threads of the computation program to this core number. However, if the user wants to keep some cores for other purposes, the number of threads can be reduced by on the corresponding button (drop down menu shown below).



7. The next setting (`Maximum log level`) is for debugging pupose and/or signalling a bug (from 1=low information level to 4=high information level).

8. The graphic interface memorizes recently opened projects. The edit field is used to set the maximum of memorized recent projects.

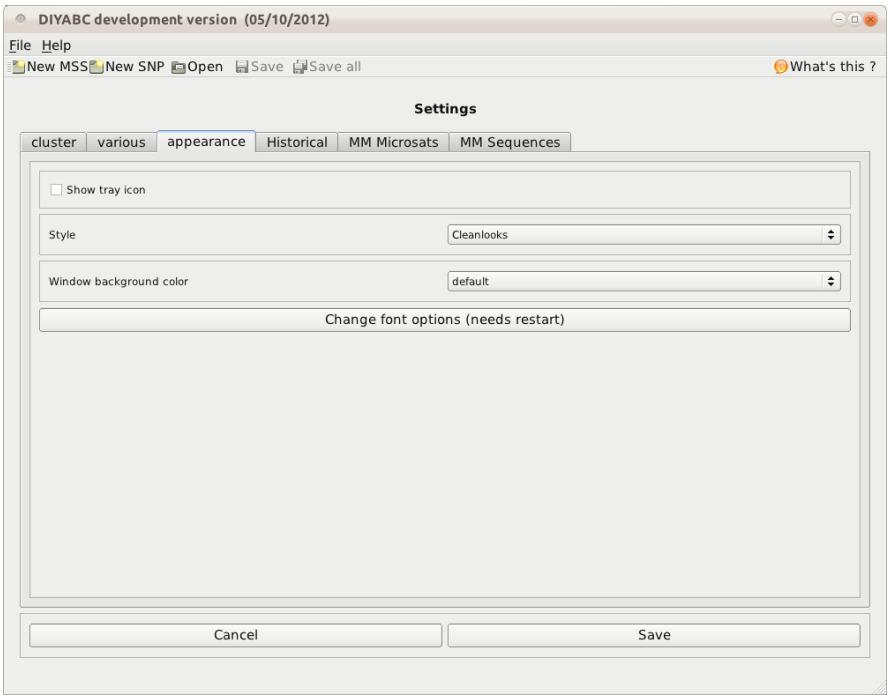
9. The last setting concerns the format of graphic files output by different analyses. Choice is shown below:



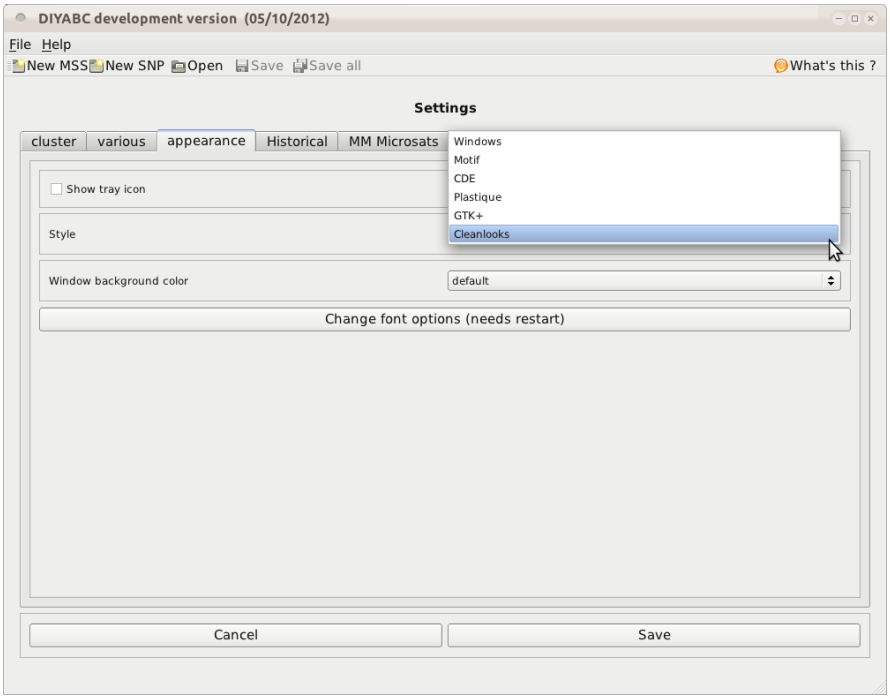
Eventually, if changes have been made, they can be either saved or cancelled (two bottom buttons).

3.7.2 Tab "appearance"

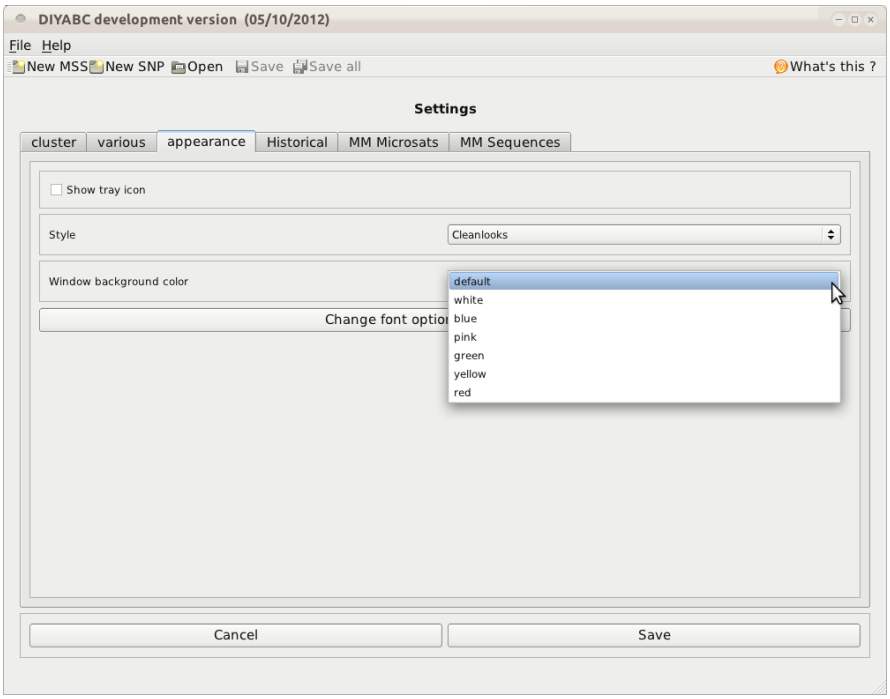
Clicking on this tab results in the following screen :



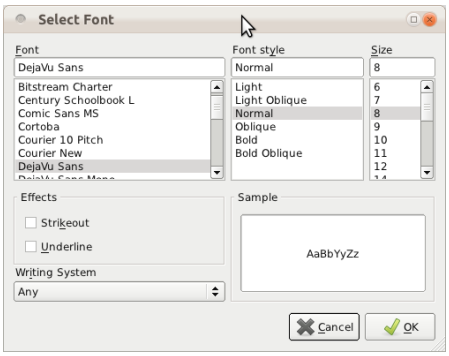
The window style can be chosen among the following (click on the upper drop down menu) :



Likewise, the background color can be chosen among the following colours :



Eventually, one can change the font of texts appearing in the different windows by clicking on the corresponding button. A usual font menu then appears allowing the desired change :



3.7.3 Tab "cluster"

The third tab is related to the use of a computer cluster to perform computations of the reference table. If you have access to a computer cluster and if the computer cluster runs a scheduler queuing system, then you can use it to generate the reference table (detailed in section 5). You will need to :

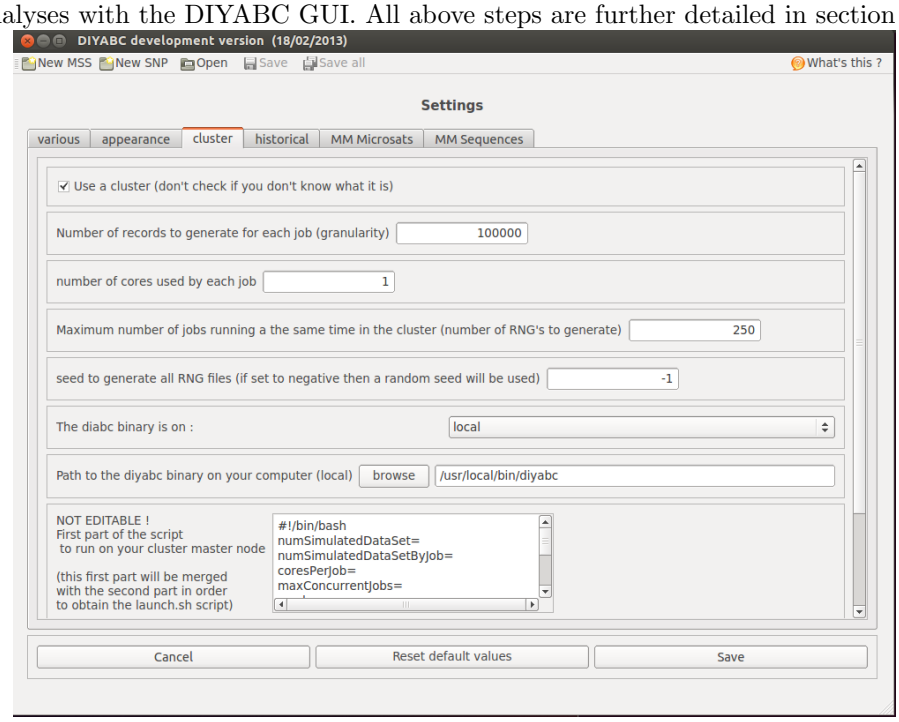
1. check the box Use a cluster (...)
2. indicate the number of data sets produced by each single job of the queue
3. indicate the number of cores used by each single job of the queue
4. indicate the number of concurrent jobs running at the same time on the cluster
5. indicate the seed to start the generation of RNG files. Leave it blank or write `None` to use a random seed

The next two text frames deals with first and last parts of the main script running on the cluster. This bash script will submit jobs to the scheduler queuing system :

1. the first part is not editable as it include the variables used by DIYABC GUI frontend

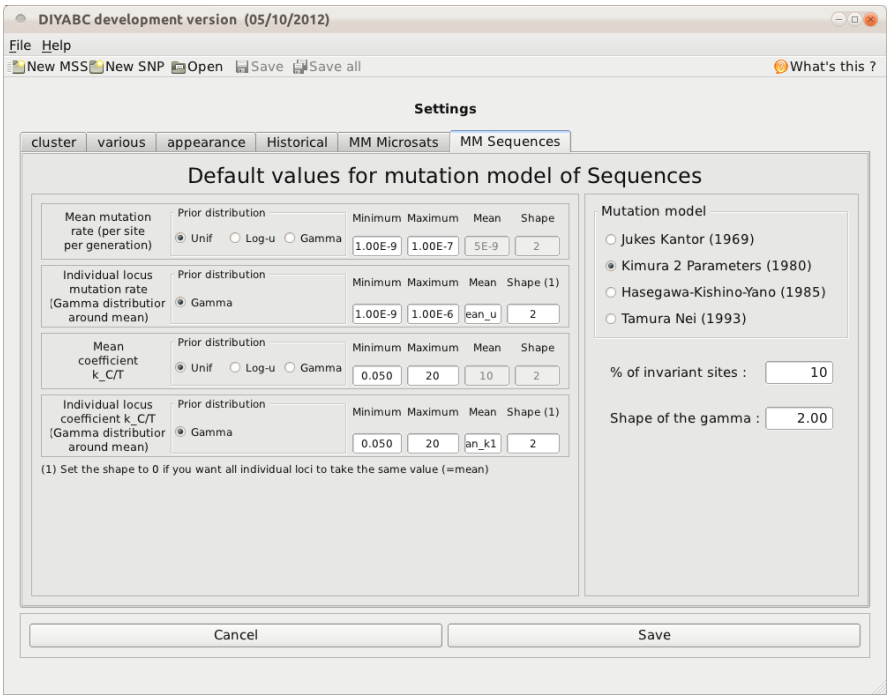
2. the last part deals with the jobs submission. You can edit it to match the specification of your scheduler : submission syntax, queue, ... By default, the code targets a Grid Engine cluster. Please ask for help to your cluster system administration.

Clicking on the Run computation button generates a bundle (*i.e.* a set of zipped files) including all you need to generate your reference table. You need to transfer the bundle in your cluster account and run it. Once all conputations are done and all the reference table parts are merged in one, you have to transfert the merged reference table back to your DIYABC project on your own computer to proceed subsequent analyses with the DIYABC GUI. All above steps are further detailed in section 5..



3.7.4 Tabs "MM Microsats" and "MM Sequences"

These two tabs are used to modify the default values of mutation parameters (MM means Mutation Model), for microsatellites and DNA sequences respectively. As an example, here is the screen corresponding to the tab "MM Sequences" :



The initial default values have been obtained through literature compilation and are valid for a large number of species. However, some species may have values that differ substantially from most species. For instance, the mutation rate of some *Drosophila* species are much lower than the values encountered in many other species (Schug *et al.*, 1997; V'azquez *et al.*, 2000) and is outside the range indicated in the initial default values.

4. Implementation details

4.1 Software design

DIYABC v2 has been designed in a very different way compared to version 1. Version 1 was a single executable file were the GUI [5] and computation codes were highly intricated and both written in the same language (*Delphi*). In version 2, the GUI and the computation codes have been completely separated. Actually, the GUI is a script written in *python* and all computations are included in a program written in *C++*. In opposition to *Delphi* which is restricted to a single OS (`Windows`), *python* and *C++* can be used with the main three OS (`Linux`, `Mac` and `Windows`), allowing version 2 to be operated under all three OS.
The GUI uses the *Qt* graphic library. The computation code is linked to the *openmp* library allowing a better use of multicore/multiprocessor computers.
The GUI can launch the computation program with the right parameters and keeps track of the progress of the latter through small log files. The GUI can launch as many computation programs as there are open projects, but no more than one computation program per project. A *lock* file located in the project directory is created when the computation program is launched by the GUI and removed when the computation program has normally terminated. When the computation program has exited anormaly, the GUI issues an error message trying to explain where the programm failed.

4.2 Files

The program uses and produces various files which we will describe now.

4.2.1 data files

Data files are text files that contain information about the samples : number and names of microsatellite markers, multilocus genotypes of individuals. The basic format is that of the Genepop software (Raymond and Rousset, 1995) and data files produced by DIYABC are under this format. **Microsatellite genotypes must be noted with 3 (haploid) or 6 (diploid) digits, these three digit numbers being the length in nucleotides of the corresponding PCR products**. In addition, we have added some features to this basic format in order to use sequence data. All these additions are explained in section 4.4. SNP data correspond to a different file format, also detailed in section 4.4.
Any extension is accepted for datafile names, including no extension at all. If the data file is simulated with DIYABC, the extension is `mss` for microsatellite/DNA sequence data and `snp` for SNP data. The next page shows examples of data sets saved.

4.2.2 reference table files

Reference table files are binary files which include two successive parts :

- The first part is a header which contains information necessary to read the second part, such as the number of scenarios, or the number of parameters of each scenario.
- The second part contains simulated data set records, each record containing the scenario number, the parameter and summary statistics values.

Each time a reference table is created or increased (each time the Run computation button is pressed), a text file is created in the project directory with the name `first_records_of_the_reference_table_X.txt` in which `X` is an integer number starting at 0 and increasing each time the Run computation button is pressed. This file provides a text version of the first n newly created records of the reference table (n being equal to the *Particle loop size*, see section 3.7.3).

4.2.3 output files

As already seen, DIYABC achieves different analyses : comparison of scenarios, estimation of posterior distribution of parameters, model checking, computation of bias and mean square errors and evaluation of confidence in scenario choice. Each analysis has its own output which can be printed and saved. Graphs

[5] Graphic User Interface

are saved under the chosen format and non-graphic output are saved in text files.

We now describe all the files produced by each type of analysis. These files are located in directories (one directory per analysis) gathered in the `analysis` subdirectory of the project directory. Below is an example of the `TOYTEST2_2012_9_26-1` project directory substructure:

```
├── TOYTEST2_2012_9_26-1
│   └── analysis
│       ├── bias1-1_bias
│       ├── bias1-2_bias
│       ├── bias1-3_bias
│       ├── bias1-4_bias
│       ├── bias1-5_bias
│       ├── bias1_bias
│       ├── compscen_comparison
│       ├── conf1_confidence
│       ├── conf2_confidence
│       ├── estim_s2_estimation
│       ├── mc_scen2_modelChecking
│       ├── mc_scen2_newmsstat-1_modelChecking
│       ├── mc_scen2_newmstat-1_modelChecking
│       ├── mc_scen2_newmstat-2_modelChecking
│       ├── mc_scen2_newmstat-3_modelChecking
│       ├── mc_scen2_newmstat-4_modelChecking
│       ├── mc_scen2_newmstat-5_modelChecking
│       ├── mc_scen2_newmstat_modelChecking
│       ├── mc_scen2_newsstat_modelChecking
│       ├── new4_modcheck_stats_-1_modelChecking
│       ├── new4_modcheck_stats_-2_modelChecking
│       ├── new4_modcheck_stats_-3_modelChecking
│       ├── new4_modcheck_stats_-_et_+_modelChecking
│       ├── NEW_NEW_TEST_MODCHECK_modelChecking
│       └── preval_pca
│           └── pictures
```

Note that each directory name starts with the name of analysis followed by the type of analysis, *e.g.* `bias` for a bias/precision analysis or `comparison` for a comparison of scenarios. In addition, when a picture has been saved, the corresponding file is located under a subdirectory named `pictures` (*e.g.* at the bottom of the figure above).

Pre-evaluate scenario prior combinations : This analysis can produce two output files named `ACP.txt` and `locate.txt`. The former is the output of the Principal Component Analysis and the latter that of the analysis giving the proportion of simulated data sets which have a value below the observed value for every summary statistics. This latter file is exactly what appears in the GUI. The structure of the `ACP.txt` file is the following. The first line indicates the number of points of the PCA, the number of PCA components (axes) and the inertia of each component, all values are separated by a single space. The second line provides the components of the observed data. It starts with a zero which corresponds to the scenario number in the following lines. Each subsequent line provides the components of data simulated according to a given scenario which number is at the beginning of the line. If one or more PCA figures have been saved, the corresponding files are saved in the `pictures` subdirectory. They are named as `refTable_PCA_X_Y_N.pdf`, with `X` and `Y` giving the axis numbers and `N` being the number of represented points.

Compute posterior probabilities of scenarios : This analysis produces three output text files : `compdirect.txt`, `complogreg.txt` and `compdirlog.txt`. The latter is directly visualized in the GUI when clicking the view numerical results button. The first two files are used by the GUI to elaborate the two graphics (Direct approach and Logistic regression). Again, if graphics have been saved, the corresponding file(s) is(are) in the `pictures` subdirectory of the analysis directory.

Evaluate confidence in scenario choice : This analysis produces a single output file, `confidence.txt`, the content of which is visualized in the GUI.

Estimate posterior distributions of parameter : Nine files are written as output of this type of analysis :

- three files `mmmq_original.txt`, `mmmq_composite.txt` and `mmmq_scaled.txt` contain the statistics (mean, median, mode and quantiles) for the original, composite and scaled parameters, respectively. They are visualized in the GUI when clicking the view numerical results button.
- three files `paramstatdens_original.txt`, `paramstatdens_composite.txt` and `paramstatdens_scaled.txt` are used by the GUI to produce the graphics showing prior/posterior distribution.
- three files `phistar_original.txt`, `phistar_composite.txt` and `phistar_scaled.txt` contains the ϕ^* values of the original, composite and scaled parameters, respectively. These files can be used for instance to redraw posterior distributions, *e.g.* with the *R* software.

As already mentionned, saved graphics are located in a `pictures` subdirectory.

Compute bias and precision of parameter estimations : Three files `bias_original.txt`, `bias_composite.txt` and `bias_scaled.txt` are produced by this type of analysis. All three files are visualized in the GUI.

Perform model-checking The output files of this type of analysis are the same as those of the *Pre-evaluate scenario prior combinations* analysis (see above). The only difference is in the names of the two text files which start with `mc` for `model checking`.

In addition, the GUI program writes several files in the project directory :

command.txt : this text file contains the history of commands issued by the GUI to be achieved by the computation program.

conf.analysis : this text file contains information about analyses.

conf.gen.tmp : this text file contains information about the loci, the genetic parameters and the summary statistics.

conf.hist.tmp : this text file contains information about the scenario and the historical parameters.

conf.th.tmp : This text file contains the title line of the reference table.

conf.tmp : This text file contains the name of the dataset and the number of parameters and summary statistics.

header.txt : This text file is a concatenation of the previous four files and is red by the computation program.

xxx.DIYABCproject : This text file contains the path to the `xxx` project.

RNG$_state_0$000.*bin* : This binary file contains the current state of the random generator.

init$_rng.out$: This text file contains information about the initialization of the random generator.

The computation program writes the following files in the project directory :

reftable.log : This text file is produced when a reftable is increased. It provides the GUI with information about the progress of computations : achieved number of records, time left.

statobs.txt : This text file is written every time an analysis is performed. It contains the values of summary statistrics for the observed data set.

The following files are output by the computation program everytime it has been launched by a specific command of the GUI (their use is only for debugging purposes and they are all in the project directory) :

general.out : when computing a reftable.

pre-ev.out : when performing a *Pre-evaluate scenario prior combinations* analysis.

compare.out : when performing a *Compare scenarios* analysis.

confidence.out : when performing a *Confidence in scenario choice* analysis.

estimate.out : when performing a *ABC parameter estimation* analysis.

bias.out : when performing a *bias-precision* analysis.

modelChecking.out : when performing a *model checking* analysis.

When performing a *Bias-precision* or a *Confidence in scenario choice* analysis, the computation program simulates what we call *pseudo-observed datasets*. The parameter and summary statistics values of these pseudo-observed datasets are written in a text file named **pseudo-observed_datasets_xxx.txt** in which **xxx** is the name given to the analysis.

4.3 Missing data

Missing or undetermined genotypes should be coded as `000` (haploid microsatellites), `000000` (diploid microsatellites), < [ ] > (haploid sequences) or < [ ][ ] > (diploid sequences) and `9` (SNP) in the data file.
Missing data are taken into account in the following way. For each appearance of a missing genotype in the observed data set, the programs records the individual and the locus. When simulating data sets, the program replaces the simulated genotype (obtained through the coalescence process algorithm) by the missing data code at all corresponding locations. All summary statistics are thus computed with the same missing data as for the observed data set.

WARNING: datafiles with virtually any amount of missing data can be analysed by DIYABC. However, for each locus a minimum of one genotyped individual per population is required. This is because summary statistics cannot be computed at a given locus in a given population if only missing data are present.

4.4 Data files

There are two different incompatible formats for data files, one for SNP loci and the other for microsatellite/DNA sequence data.
For the microsatellite/DNA sequence data, the format already presented in version 1 of DIYABC is an extended Genepop format. The additional features are :

1. In the title line appears the sex ratio noted between < and > under the form $< NM = rNF >$, in which r is the ratio of the number of females per male (*e.g.* $< NM = 2.5NF >$ means that the number of males is 2.5 times the number of females; for a balanced sex ratio one should write $< NM = 1.0NF >$). Since the title is generally only copied, this addition should not interfere with other programs using Genepop datafiles. Also if there is no such sex ratio addition, DIYABC will consider by default that NM=1.0NF.

2. After the locus name, there is an indication for the category of the locus which is $< A >$ for autosomal diploid loci, $< H >$ for autosomal haploid loci, $< X >$ for X-linked (or haplo-diploid) loci, $< Y >$ for Y-linked loci and $< M >$ for mitochondrial loci. If no category is noted, DIYABC will consider the locus as autosomal diploid or autosomal haploid depending on the corresponding genotype of the first typed individual.

3. Genotypes of microsatellite loci are noted with six digit numbers (e.g. 190188) if diploid and by three digit numbers (e.g. 190) if haploid.

4. Sequence locus are noted between < and > . In addition each sequence alleles/haplotypes is noted between brackets. For instance, a haploid sequence locus will be noted < [GTCTA] > and a diploid sequence locus < [GTCTA][GTCTT] >. Sequences may contain undetermined nucleotides which will be denoted Â N Â or Â - Â. Note that all sequence alleles/haplotypes have to be of similar length. The length of shorter sequence allele/haplotypes needs to be adjusted to the larger sequence allele/haplotype by adding Â N Â or Â - Â symbols at the end of the sequences. It is worth stressing that, aT a given locus, only the portion of the sequence shared by all individuals of the dataset will be used for computing summary statistics. We therefore advise removing locus-individual sequence data with too many "N" and replace them by missing data. Finally, remember that this version of

the program does not consider insertion-deletion mutations, mainly because there does not seem to be much consensus on this topic.

5. Missing microsatellite genotypes are noted 000 if haploid or 000000 if diploid.

6. Missing sequence alleles/haplotypes are noted < [ ] > if haploid or < [ ][ ] > if diploid.

For *SNP data*, the datafile format includes:

- a first line (headline) providing the sex-ratio as above (the e.g.$< NM = 1.0NF >$for a balanced sex ratio), the required MAF (minimum allele frequency criterion; e.g. <MAF=0.05> or <MAF=hudson>), and any text that can be used as a title . Information on the sex ratio and the MAF can be anywhere in this line. The MAF is computed pooling all genes genotyped over all studied population samples. For instance, the specification of a MAF equal to 5% (i.e. <MAF=0.05>) will automatically select a subset of m loci characterized by a minimum allele frequency > 5% among the l locus of the observed dataset. In agreement with this, only m locus with a MAF>5% will be retained in a simulated dataset (simulated loci with a MAF≤5% will be discarded). Writing <MAF=hudson> (or omitting to write any instruction with respect to the MAF) will bring the program to use the standard Hudson's algorithm without further selection as done so far in the previous version of DIYABC.

- a second line starting with the three keywords `IND SEX POP`, separated by at least one space, followed by as many letters as SNP loci, the letter giving the location of the locus as above ($< A >$ for autosomal diploid loci, $< H >$ for autosomal haploid loci, $< X >$ for X-linked (or haplo-diploid) loci, $< Y >$ for Y-linked loci and $< M >$ for mitochondrial loci). Letters are separated by a single space.

- as many lines as there are genotyped individuals, with the code-name of the individual, a letter (M or F) indicating its sex, a code-name for its population and the values (0, 1 or 2) of the number of the (arbitrarily chosen) reference allele at each SNP locus. For instance in the case autosomal diploid SNP loci, we have 0 = homozygous genotype for the non reference allele, 1 = heterozygous genotype for the reference allele, 2 = homozygous genotype for the reference allele. It is worth noting that for autosomal haploid loci (denoted H), as well as for mitochondrial loci (denoted M) and Y-linked loci (denoted Y), the SNP genotypes will be 0 or 1.

- Only a subset of the SNP loci included in the data file can be considered (selected) in the simulations and hence in subsequent ABC analyses. For instance one can choose to select in the corresponding panel the SNP loci 1 to 1000 of a data file including a total of say 10000 loci. This allows running faster simulations and processing independant replicate ABC analyses of sets of 1000 SNP loci by considering loci 1 to 1000 and then 1001 to 2000, and so on, in separate analyses.

- Following Hudson's (2002) criterion, only polymorphic SNP loci (over the entire dataset) are considered. Monomorphic SNP loci (over the entire dataset) are automatically filtered by the program. It is preferable, however, that the user removes himself all monomorphic loci from his/her (observed) dataset before submitting it to DIYABC.

- Before running any simulation, DIYABC provides a text file including the set of SNP loci selected from the observed dataset (e.g. polymorphic loci 1 to 1000 with a MAF=0.05). This file is named "UserDataFileName.bin.txt".

Below are three examples of data sets that can be analyzed with DIYABC.

In the *first example*, this data set includes two population samples, each of 12 diploid individuals (8 females and 4 males in the first sample and 5 females and 7 males in the second sample). As deduced from the letter between < and > on the locus name lines (see page 25), these individuals have been genotyped at 3 microsatellite loci (1 autosomal $< A >$, 1 X-linked $< X >$ and 1 Y-linked $< Y >$) and 3 DNA sequence loci (1 autosomal. 1 X-linked and 1 mitochondrial $< M >$). The species sex-ratio, given in the title line, is of three males for one female ($< NM = 3NF >$) or in other words, the number of males equals three times the number of females.

```
Data file example <NM=3NF>
A107   <A>
A1023  <X>
A1101  <Y>
S001   <A>
S019   <X>
S025   <M>
Pop
1-01 ,    172176 156172 000   <[TAGCTA][TAGCTA]>  <[GATCG][GATCG]>   <[ATTACTTA]>
1-02 ,    180204 156184 000   <[TAGCTA][TAGCTA]>  <[GATCG][GATCG]>   <[ATTACTTG]>
1-03 ,    150162 168192 000   <[TAGCTA][TAGCTA]>  <[GATCG][GATCG]>   <[ATTACTTG]>
1-04 ,    210218 158182 000   <[TAGCTA][TAGCTA]>  <[GATCG][GATCG]>   <[ATTACTTA]>
1-05 ,    188218 154168 000   <[TAGCTA][TAGCTA]>  <[GATCG][GATCG]>   <[ATTACTTG]>
1-06 ,    180190 160152 000   <[TAGCTA][TAGCTA]>  <[GATCG][GATCG]>   <[ATTACTTA]>
1-07 ,    168208 172216 000   <[TAGCTA][TAGCTA]>  <[GATCG][GATCG]>   <[ATTACTTA]>
1-08 ,    174168 160202 000   <[TAGCTA][TAGCTA]>  <[GATCG][GATCG]>   <[ATTACTTG]>
1-09 ,    176162 180    316   <[TAGCTA][TAGCTA]>  <[GATCG]>          <[ATTACTTA]>
1-10 ,    150194 220    296   <[TAGCTA][TAGCTA]>  <[GATCG]>          <[ATTACTTA]>
1-11 ,    158176 182    326   <[TAGCTA][TAGCTA]>  <[GATCG]>          <[ATTACTTG]>
1-12 ,    154156 166    318   <[TAGCTA][TAGCTA]>  <[GATCG]>          <[ATTACTTA]>
Pop
2-01 ,    168164 184216 000   <[TAGCTA][TAGCTA]>  <[GATCG][GATCG]>   <[ATTACTTG]>
2-02 ,    164160 152208 000   <[TAGCTA][TAGCTA]>  <[GATCG][GATCG]>   <[ATTACTTG]>
2-03 ,    166222 180170 000   <[TAGCTA][TAGCTA]>  <[GATCG][GATCG]>   <[ATTACTTG]>
2-04 ,    212150 228166 000   <[TAGCTA][TAGCTA]>  <[GATCG][GATCG]>   <[ATTACTTA]>
2-05 ,    210152 228196 000   <[TAGCTA][TAGCTA]>  <[GATCG][GATCG]>   <[ATTACTTG]>
2-06 ,    156212 178    292   <[TAGCTA][TAGCTA]>  <[GATCG]>          <[ATTACTTG]>
2-07 ,    166222 174    302   <[TAGCTA][TAGCTA]>  <[GATCG]>          <[ATTACTTA]>
2-08 ,    196200 168    278   <[TAGCTA][TAGCTA]>  <[GATCG]>          <[ATTACTTG]>
2-09 ,    174174 172    292   <[TAGCTA][TAGCTA]>  <[GATCG]>          <[ATTACTTA]>
2-10 ,    178194 212    282   <[TAGCTA][TAGCTA]>  <[GATCG]>          <[ATTACTTG]>
2-11 ,    204160 180    304   <[TAGCTA][TAGCTA]>  <[GATCG]>          <[ATTACTTG]>
2-12 ,    190226 160    226   <[TAGCTA][TAGCTA]>  <[GATCG]>          <[ATTACTTG]>
```

In the *second example*, the species is haploid. Individuals have been genotyped at three autosomal microsatellite loci and one mitochondrial DNA sequence locus. The species being haploid (deduced from the presence of autosomal haploid loci), no indication of the sex-ratio appears in the title line.

```
Data file example
B153   <H>
B632   <H>
B046   <H>
COI    <M>
POP
  And-001 ,   164   184   210   <[TCCTTCCGTTGTGCGACCACTTCGTACGTT]>
  And-002 ,   164   182   214   <[TCCTTCCGTTGTGCGACCACTTCGTACGTT]>
  And-003 ,   150   186   214   <[TCCTTCCGTTGTGCGACCACTTCGTACGTT]>
  And-004 ,   160   188   220   <[TCCTTCCGTTGTGCGACCACTTCGTACGTT]>
  And-005 ,   152   176   214   <[TCCTTCCGTTGTGCGACCACTTCGTACGTT]>
POP
  Bor-001 ,   154   188   210   <[TCCTTCCGTTGTGCGACCACTTCGCACGTT]>
  Bor-002 ,   154   196   206   <[TCCTTCCGTTGTGCGACCACTTCGCACGTT]>
  Bor-003 ,   166   194   202   <[TCCTTCCGTTGTGCGACCACTTCGCACGTT]>
  Bor-004 ,   150   194   200   <[TCCTTCCGTTGTGCGACCACTTCGCACGTT]>
POP
  Cam-001 ,   202   222   202   <[TCCTTCCGTTGTGCGGCCACTTCGTACGTT]>
  Cam-002 ,   226   206   198   <[TCCTTCCGTTGTGCGGCCACTTCGTACGTT]>
  Cam-003 ,   216   206   208   <[TCCTTCCGTTGTGCGACCACTTCGTACGTT]>
```

In the *third example*, the species is diploid and was genotyped at 23 SNP loci: 20 autosomal loci, 1 X-linked locus, 1 Y-linked locus and 1 mitochondrial locus. The first line provides the title which includes the species sex-ratio and the MAF (minimum allele frequency). The second line indicates: individual name in column 1, individual sex in column 2 (`M` for male, `F` for female, `9` or any other letter if unknown), population name in column 3 and one column per SNP locus (letter `A` for an autosomal locus, `X` for an X-linked locus, `Y` for a Y-linked locus and `M` for a mitochondrial locus). Columns are separated by one or more spaces. SNP genotypes are coded `0`, `1` or `2` (`9` for missing data) according to the number of reference alleles at the corresponding locus. Note that the sex has no influence on simulations for autosomal, mitochondrial or haploid loci (any sex can be hence declared). For individuals with an unknown sex (denoted `9`, see `IND P1_2, P1_3 and P2_15)`, data for autosomal (as well as mitochondrial and haploid) loci will be taken into account and simulated. On the other hand, the genotypes of X-linked and Y-linked loci for the `same IND P1_2, P1_3 and P2_15` with unknown sex cannot be safely determined and are hence noted `9` for missing data.

```
Example Datafile for SNP <NM=1.5NF>  <MAF=hudson>
IND   SEX   POP    A A A A A A A A A A A A A A A A A A A X Y M
P1_1   F    P1     0 0 0 1 1 1 0 0 0 1 1 0 1 0 9 0 0 0 0 0 0 9 1
P1_2   9    P1     1 0 0 1 0 0 0 0 0 1 2 0 0 0 0 0 0 1 0 0 9 9 0
P1_3   9    P1     0 1 0 0 1 0 0 0 0 1 2 0 1 0 0 0 0 0 1 0 9 9 1
P1_4   F    P1     0 0 9 1 2 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 9 1
P1_5   F    P1     0 0 9 0 1 2 0 0 1 1 2 0 0 0 0 0 1 0 0 0 0 9 0
P1_6   F    P1     0 0 1 0 2 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 9 1
P1_7   F    P1     0 0 0 1 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 9 0
P1_8   F    P1     0 0 1 0 1 0 0 0 0 0 2 0 0 0 0 0 0 9 1 0 0 9 1
P1_9   F    P1     0 0 1 0 0 0 0 0 0 2 2 0 0 0 0 0 0 0 0 0 0 9 1
P1_10  F    P1     0 0 0 0 0 0 0 0 0 1 2 0 0 0 0 0 0 0 0 0 0 9 1
P1_11  M    P1     0 1 0 0 0 1 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 1
P1_12  M    P1     0 0 0 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 1 1 1
P1_13  M    P1     0 0 0 1 0 0 0 0 0 2 2 0 1 0 0 0 1 0 0 1 0 0 1
P1_14  M    P1     0 1 0 0 0 1 0 0 0 0 2 0 0 0 0 0 0 0 2 0 0 1 1
P1_15  M    P1     1 0 0 0 1 1 1 0 0 1 2 0 0 0 1 0 0 0 1 0 0 0 1
P2_1   F    P2     0 0 0 1 0 0 0 0 0 0 2 0 0 0 1 0 0 0 2 0 0 9 1
P2_2   F    P2     0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 9 1
P2_3   F    P2     0 0 0 0 0 0 0 0 0 0 2 0 1 0 0 0 0 0 1 0 0 9 1
P2_4   F    P2     0 0 0 0 1 0 0 0 0 0 2 0 0 0 0 0 0 0 1 1 1 9 1
P2_5   F    P2     0 0 0 1 2 9 0 0 0 2 2 1 0 0 0 0 0 0 1 0 0 9 1
P2_6   F    P2     0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 0 0 2 1 0 9 1
P2_7   F    P2     0 0 0 1 1 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 9 1
P2_8   F    P2     0 0 1 1 0 0 0 0 0 2 1 0 0 0 0 0 0 0 0 0 0 9 0
P2_9   F    P2     0 0 0 2 0 0 0 0 2 0 2 0 0 0 0 0 0 0 2 0 0 9 1
P2_10  F    P2     1 0 0 0 1 0 0 0 0 1 2 0 0 0 0 0 0 0 1 0 0 9 1
P2_11  F    P2     0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 9 1
P2_12  F    P2     0 0 0 0 1 0 0 0 0 0 2 0 0 0 0 1 0 0 0 0 0 9 1
P2_13  M    P2     0 0 0 0 1 1 0 0 0 1 2 0 0 1 0 0 0 0 1 0 0 0 1
P2_14  M    P2     0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0
P2_15  9    P2     0 0 0 1 0 0 0 1 0 2 2 0 0 0 0 0 0 0 0 0 9 9 0
```

5. Cluster version

The ABC method requires simulating many data sets, which is time consuming. Typically, one to several millions data sets are needed to build up an interresting reference table and this process can last several hours to several days. Hence it might be useful to take advantage of a computer grid cluster.

This part of the notice describe how to use a cluster with the GUI frontend in Section 5.1. Advices to distribute the workload in jobs on the cluster are given in Section 5.2. For advanced users who need more information, Section 5.3 describes the jobs that are sent to the queueing system of the cluster, and Section 5.4 sums up how DIYABC produces independent random number generators (RNG's).

5.1 Using a cluster with DIYABC

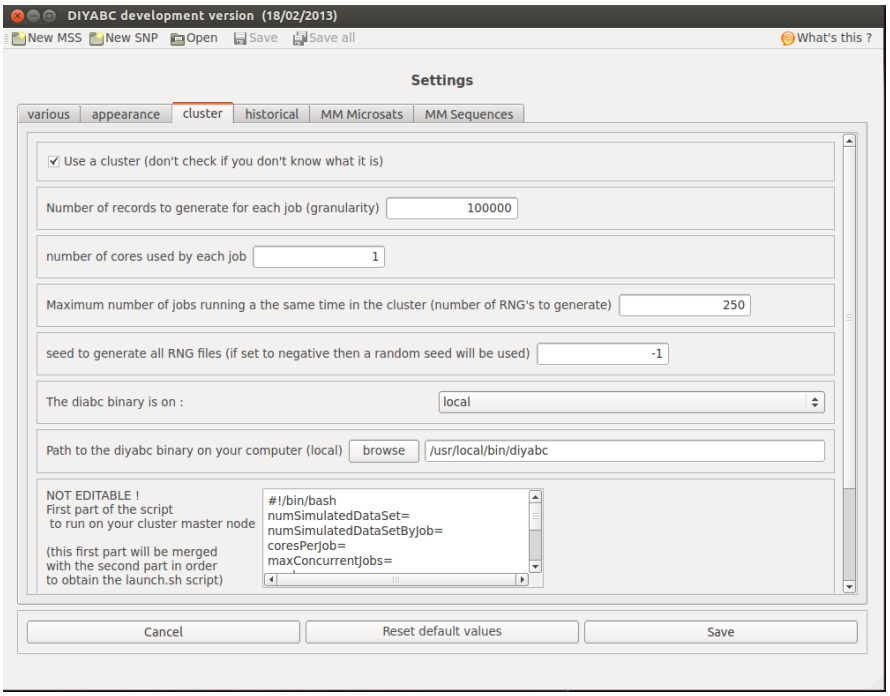
You can prevent DIYABC from simulating data sets on your own computer by checking use a cluster in the setting panel of the GUI. Then, instead of computing the simulations on your computer, the GUI frontend will prepare a bundle for the cluster. Note that, while this option remains checked, DIYABC will not compute any reftable on your computer.
Generating a reference table with a cluster is a three stage process.

1st stage Configure the required parameters in the GUI frontend and generate the cluster bundle (set of zipped files).

2nd stage Transfer the bundle to the cluster and run it.

3rd stage Transfer back the reference table and include it to the project

In the cluster tab of the settings panel, you can configure the useful parameters to send correct orders to the job scheduler of your cluster. The bash script named `launch.sh` of the cluster bundle produced by the GUI will sent those orders to the job scheduler of the cluster. To be able to write this bash script, the GUI needs informations on our cluster you can give by filling the fields of the cluster tab.



5.1.1 Configuring distribution of the workload on the cluster

You have to understand and edit six parameters of the cluster settings. We shall recall here that the "DIYABC binary" program is able to take advantage of multi-core computers. Thus to divde the whole workload, the cluster bundle can run several jobs, and configure each job to use several CPU cores of a

given machine in the cluster[6]. The six parameters might be given by filling the blank fields of the cluster tab, and are as follow.

- "*Number of simulations oer job (granularity)*" or `numSimulatedDatasetByJob`: This integer indicates the number of data sets simulated by each single job. It represents the granularity of your computations on the cluster.
- "*Number of cores per job*" or `coresPerJob`: This integer indicates how many CPU cores will be used by each job.
- "*Maximum number of jobs running at the same time in the cluster*" or `maxConcurrentJobs`: This integer indicates the maximum number of jobs allowed to run simultaneously on the cluster.
- "*First seeds of the RNG's*" or `seed`: This integer indicates to DIYABC the seeds to initialize the RNG's. By writing `-1` you ask the program to use random seeds (recommended). Starting from user-defined seeds is mainly for testing or debbugging purposes.
- "*The DIYABC binary is on*": This option determines whether the DIYABC binary program (`general`) is already installed on the cluster (then set `cluster`) or if the bundle has to import a suitable binary (then set `local`).
- "*Path to the DIYABC binary*" or `DIYABCPath`: This string indicates the path of a DIYABC binary program that can run on the machines of the cluster. Note that the binary can be on your own computer (you can browse your file system to choose the correct DIYABC binary with the dedicated button) or on your cluster, in which case it is highly recommanded to specify an absolute path.

5.1.2 Dealing with the job scheduler of the cluster

The two last text boxes of the cluster tab in the settings panel deal with the main script `launch.sh` executed on the master node of the cluster. This script generates a pool of RNG files, submits the jobs to the scheduler queuing system using a `node.sh` script, monitors the jobs and merges all reftable files into a single reftable file when all jobs are completed. The last box (which is the only one that be edited) deals with the jobs submission. By default, `launch.sh` targets a *Grid Engine* cluster. You probably need to customise this script to fit your cluster configuration (scheduler system, queue name, ...). You should mainly need to modify the `##### EDIT #####` section to comply with the rules of the job queueing system of the cluster. Please ask for help to your cluster system administrator.

5.1.3 Transfer the bundle to the cluster and run it

Once you have checked the box Use a cluster (...), configured the cluster parameters in the settings panel and saved them, you need to click on the Run computation button from your project panel. The program will ask you the name of the tar archive you will have to copy on the cluster to run the computations. This tar archive can be copied to your cluster account by many ways (for instance with the help an sftp client like FileZilla or WinSCP). Once the archive is on your cluster working directory, you can log in your cluster account with a shell console and untar your archive by typing :

```
tar -xvf <yourTarArchiveName.tar>
cd yourTarArchiveName
```

This will create a directory with all the files needed to run DIYABC, namely

1. general : the DIYABC binary program or executable
2. header.txt : the header file
3. launch.sh : the main script to run
4. node.sh : the script that will be runned by your scheduler for each job
5. $< yourData.mss >$: the data file

[6]The multi-threaded capacity of DIYABC was programmed with the OpenMP API. This means that DIYABC can use several cores in a single computer but, contrary to MPI-based program, a single job cannot use several cores distributed on different computers. So please be sure to use an apropriate parallele environment to submit your jobs. Ask to your cluster system administrator.

You can now run the main script by typing `./launch.sh` in the shell console. If everything was set correctly, you can monitor the progression of the computations in the console. For instance, for a total of 50,000 data sets to be produced through 5 jobs of 10,000 data sets:

, numbers=left, numberstyle=, stepnumber=1, numbersep=5pt, backgroundcolor=
, tabsize=4, captionpos=b, breaklines=true, breakatwhitespace=false, title=, keywordstyle=, commentstyle=, stringstyle=,
escapeinside=%**), morekeywords=*, deletekeywords=

```
>launch.sh
** Generation of RNG files :
./general -p ./ -n "t:1;c:5;s:1038"

** jobs submition :

qsub -N n1_test -q short.q -cwd node.sh 10000 /home/dehneg/DIYABCtest 1
test.mss
Your job 111598 ("n1_test") has been submitted

qsub -N n2_test -q short.q -cwd node.sh 10000 /home/dehneg/DIYABCtest 2
test.mss
Your job 111599 ("n2_test") has been submitted

qsub -N n3_test -q short.q -cwd node.sh 10000 /home/dehneg/DIYABCtest 3
test.mss
Your job 111600 ("n3_test") has been submitted

qsub -N n4_test -q short.q -cwd node.sh 10000 /home/dehneg/DIYABCtest 4
test.mss
Your job 111601 ("n4_test") has been submitted

qsub -N n5_test -q short.q -cwd node.sh 10000 /home/dehneg/DIYABCtest 5
test.mss
Your job 111602 ("n5_test") has been submitted

** monitoring :
0/5 finished 0%  (total : 0)
1/5 finished 20% (total : 10000)
1/5 finished 20% (total : 10000)
2/5 finished 40% (total : 20000)
4/5 finished 80% (total : 40000)
5/5 finished 100% (total : 50000)

** reftables concatenation  :
./general -p /home/dehneg/DIYABCtest -q 2>&1 concat.out
************************************************************
All the result files have been concatenated into reftable.bin
See concat.out output file for logs
************************************************************
```

Once the monitoring phase starts, you can quit `launch.sh` and restart it at any time. The batch script `launch.sh` will not resubmit jobs that have already be sent to the queueing system of the cluster.

5.1.4 Transfer back the reference table and include it into your computer project

Once your final `reftable.bin` file has been produced, you need to transfer it from the cluster to your own computer (with, *e.g.*, an sftp client, see above). The `Import and merge reftable file` option from your project menu is the correct way to include the imported reference table into your project. Be careful that DIYABC do not inspect the imported reference table and do not backup the old reference table before merging them. You will not be able to recover from any error during this last stage except if you backup your old reference table.

5.2 Advices to distribute the workload on a cluster

The six parameters of the cluster settings tab allow you to optimize your use of the cluster according to your access limitations, the workload of the cluster from other users and its queueing policy. For instance, if your cluster is overloaded and if the waiting time in queue is long, then it is preferable to choose a high amount of simulations per job. On the contrary, if a queue for short jobs is free while the other queues are overloaded, then it is preferable to choose a low amount of simulations per job and to submit them to the short queue... Note also that increasing the number of cores per job generally increases the queueing waiting time. But remember that a job with 40 cores will not increase the reference table faster than 40 jobs with one core each.

Both parameters `coresPerJob` and `maxConcurrentJobs` are used to initialize a set of independent RNG's (see Section 5.4 below). One caveat of our way of producing independent RNG's is the need to simultaneously initiate all RNG's that will be used. Before starting the jobs queue submission, the main script `launch.sh` will produce of a pool of `coresPerJob` × `maxConcurrentJobs` generators, stored in `maxConcurrentJobs` "RNG files". Then, each job will randomly choose one RNG file that is not currently in use by other concurrent jobs, to initiate its own `coresPerJob` parallel RNG's and store the last states of thoses generators at the end of the computation. Be careful, if you use lots of jobs and cores simultanuously, initialisation of the RNG's will be time consuming, see Fig. 1.

number of jobs (t)	*number of cores per job (c)*						
	1	*4*	*8*	*16*	*32*	*40*	*80*
100	*20"*	*1'20"*	*3'*	*10'*	*20'*	*26'*	*30'*
200	*50"*	*3'*	*7'*	*20'*	*40'*	*50'*	*1h10'*
500	*2'*	*8'*	*25'*	*50'*	*1h40'*	*2h06'*	*2h45'*
1000	*4'*	*16'*	*50'*	*1h40'*	*2h10'*	*2h45'*	

Figure 1: ***RNG files Time Generation.*** *As shown in section 5.4, one caveat of the RNG method is the obligate generation of all the RNG files at once (generating the RNG files one by one for each job on a cluster is possible but will result in a dangerous bias). The second caveat of the RNG method is a consequence of the first one, ie the time needed to generate the RNG files increases depending on the number of RNG files* `t` *and the number of cores* `c` *available for each RNG file. Once a file is generated, it is not possible to add cores.*

5.3 A detailed description of each job

The jobs sent to the queueing system of the cluster are given in the script `node.sh`. This script can be decomposed into the following sequential stages:

1. Create a *jobid* name according to the following pattern : `<node hostname>-n-<sequential number of the job>-pid<pid of nodes.sh execution>-<a random number>` (*pid* mean Process IDentifier).

2. Use the scheduler temporary directory if the scheduler provide a `TMPDIR` environment variable or create a working temporary directory `/tmp/tmpDIYABC_<job id>` on the cluster node.

3. Choose a RNG file from the pool of RNG files created by `launch.sh`. (It means that the node must access your DIYABC *yourTarArchiveName* directory in your working directory.)

4. Run DIYABC binary program (of course !).

5. Copy periodically the reftable log file to the *yourTarArchiveName* directory. Thus, `launch.sh` can inform the user about the progress of the computations (through the total amount of simulations already performed).

Note that, as long as a job (`node.sh`) is using a RNG file, the RNG file in the *yourTarArchiveName* directory is replaced by a lock file named `<the choosen RNG file name>.lock` and a flag file named the `<choosen RNG file name>_<date of the run>_<job id>`. The flag file contain the local pid of the job on the node of DIYABC. Once a job has finished and updated the RNG file, it removes the lock and flag files and copy back the updated RNG file.

5.4 A note on the random number generators

By nature, a random number generator (RNG) is a sequential algorithm, as described in Figure 2 below. Indeed, we shall describe a RNG by its updating function f changing deterministically the internal state. Each time the user requires a new realization of the uniform distribution over $[0;1)$, the algorithm derives a value u_k from the current internal state i_k and then updates this state with f. Hence a first and important issue for parallel Monte Carlo computations is to design independent RNGs that might run in parallel while minimizing the communications between processors. It is quite standard to use as many RNGs as computing cores in the computer or in the cluster of computers.

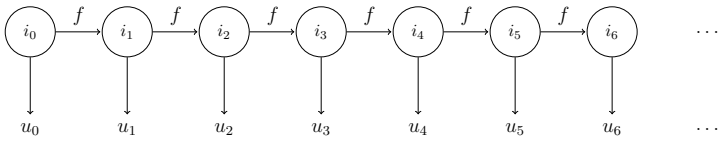


Figure 2: ***Random Number Generator.*** *A RNG is an algorithm that produces a sequence of floating numbers, says $u_0, u_1, \ldots$, that resembles a sequence of independant random numbers, uniformly distributed over $[0;1)$. It uses a sequence of internal states, say $i_0, i_1, \ldots$, which are computed by reccurence, namely, $i_{k+1} = f(i_k)$. The first internal state i_0 is often named the seed.*

The second version of DIYABC uses the Dynamic Creator (DCMT) of Matsumoto and Nishimura (2000) to look for a set of independent Mersenne-Twister generators. Actually, the updating function f of a Mersenne-Twister generator is parametrized by a few integer numbers. The output of the DCMT is a set of N updating functions, say $\{f^{(1)}, \ldots, f^{(N)}\}$, producing independent streams. That is, the n-th RNG is a sequence of iternal states $i_0^{(n)}, i_1^{(n)}, i_2^{(n)}, \ldots$ satisfying $i_{k+1}^{(n)} = f^{(n)}(i_k^{(n)})$ that gives rise to a sequence of independent, uniformly distributed numbers $u_0^{(n)}, u_1^{(n)}, u_2^{(n)}, \ldots$. We found that the DCMT was simple to use and gave good results. There is no limitation on the number N of RNGs it produces. Once initialized, the different RNGs do not require any communication between them and each of them runs as quickly as a single Mersenne-Twister generator. But an important limitation is that it is impossible to add a new RNG to the set produced by the DCMT. Practically, this means that we have to know *a priori* a bound on the number of jobs working together in parallel. See section 5.

Bibliography

Beaumont, M. A., W. Zhang and D. J. Balding, 2002. Approximate Bayesian Computation in Population Genetics. *Genetics* **162**, 2025-2035.

Beaumont, M.A., 2008. Joint determination of topology, divergence time, and immigration in population trees. In Simulation, Genetics, and Human Prehistory, eds. S. Matsumura, P. Forster, C. Renfrew. McDonald Institute Press, University of Cambridge (*in press*).

Begg, C.B. and R. Gray, 1984. Calculation of polychotomous logistic regression parameters using individualized regressions. *Biometrika*, **71**, 11-18.

Belkhir K., Borsa P., Chikhi L., Raufaste N. and F. Bonhomme, 1996-2004 GENETIX 4.05, logiciel sous Windows TM pour la genetique des populations. Laboratoire Genome, Populations, Interactions, CNRS UMR 5171, Universite de Montpellier II, Montpellier (France).

Bertorelle, G. and L. Excoffier, 1998. Inferring admixture proportion from molecular data. *Mol. Biol. Evol.* **15**, 1298-1311.

Choisy, M., P. Franck and J.M. Cornuet, 2004. Estimating admixture proportions with microsatellites : comparison of methods based on simulated data. *Mol. Ecol.* **13**, 955-968.

Chakraborty R and L Jin, 1993. A unified approach to study hypervariable polymorphisms: statistical considerations of determining relatedness and population distances. EXS. *67*, 153?175.

Cornuet, J. M., M. A. Beaumont, A. Estoup and M. Solignac, 2006. Inference on microsatellite mutation processes in the invasive mite, *Varroa destructor*, using reversible jump Markov chain Monte Carlo. *Theoret. Pop. Biol.* **69**, 129-144.

Cornuet J.M., V. Ravignï¿œ and A. Estoup, 2010. Inference on population history and model checking using DNA sequence and microsatellite data with the sofware DIYABC (v1.0). *BMC Bioinformatics* **11**,401.

Cornuet J.M., F. Santos, M.A. Beaumont, C.P. Robert, J.M. Marin, D.J. Balding, T. Guillemaud and A. Estoup, 2008. Infering population history with DIYABC: a user-friendly approach to Approximate Bayesian Computations. *Bioinformatics*, **24** (23), 2713-2719.

Cornuet, J-M., Pudlo, P., Veyssier, J., Dehne-Garcia A., Gautier M., Leblois R., Marin J-M, and A. Estoup, 2014. DIYABC v2.0: a software to make approximate Bayesian computation inferences about population history using single nucleotide polymorphism, DNA sequence and microsatellite data. *Bioinformatics*. Vol. 30, no. 8, p1187–1189, doi: 10.1093/bioinformatics/btt763.

Estoup, A., M. Solignac, M. Harry and J.M. Cornuet, 1993. Characterization of $(GT)_n$ and $(CT)_n$ microsatellites in two insect species: *Apis mellifera* and *Bombus terrestris*. *Nucl. Ac. Res.*, **21**, 1427-1431.

Estoup, A., I. J. Wilson, C. Sullivan, J. M. Cornuet and C. Moritz, 2001 Inferring population history from microsatellite et enzyme data in serially introduced cane toads, *Bufo marinus*. *Genetics*, **159**, 1671-1687.

Estoup, A., P. Jarne and J.M. Cornuet, 2002. Homoplasy and mutation model at microsatellite loci and their consequences for population genetics analysis. *Mol. Ecol.*, **11**, 1591-1604.

Estoup, A. and S. M. Clegg, 2003. Bayesian inferences on the recent islet colonization history by the bird Zosterops lateralis lateralis. *Mol. Ecol.* **12**: 657-674.

Estoup, A., M.A. Beaumont, F. Sennedot, C. Moritz and J.M. Cornuet, 2004. Genetic analysis of complex demographic scenarios : spatially expanding populations of the cane toad, *Bufo marinus. Evolution*,**58**, 2021-2036.

Estoup, A., E. Lombaert, J.M. Marin, T. Guillemaud, P. Pudlo, C.P. Robert and J.M. Cornuet, 2012. Estimation of demo-genetic model probabilities with Approximate Bayesian Computation using linear discriminant analysis on summary statistics. *Molecular Ecology Resources*, **12**, 846-855.

Excoffier, L., A. Estoup and J.M. Cornuet, 2005. Bayesian analysis of an admixture model with mutations and arbitrarily linked markers. *Genetics* **169**, 1727-1738.

Fagundes, N.J.R., N. Ray, M.A. Beaumont, S. Neuenschwander, F. Salzano, S.L. Bonatto and L. Excoffier, 2007. Statistical evaluation of alternative models of human evolution. *Proc. Natl. Acad. Sc.*, **104** : 17614-17619.

Fu, Y.X. and Chakraborty, R., 1998. Simultaneous estimation of all the parameters of a stepwise mutation model. *Genetics*, **150**, 487-497.

Garza JC and E Williamson, 2001. Detection of reduction in population size using data from microsatellite DNA. *Mol. Ecol.* **10**,305-318.

Gelman, A., J.B. Carlin, H.S. Stern and D.B. Rubin, 1995. *Bayesian Data Analysis*. Chapman et Hall, London, 526p.

Goldstein DB, Linares AR, Cavalli-Sforza LL, and Feldman MW, 1995. An evaluation of genetic distances for use with microsatellite loci. *Genetics* **139**, 463-471.

Goudet, J. ,1995. FSTAT (Version 1.2): A computer program to calculate F- statistics. *J. Hered.* **86**, 485-486.

Griffiths, R.C. and S. Tavarï¿œ, 1994. Simulating probability distributions in the coalescent. *Theor. Pop. Biol.* **46**, 131-159.

Guillemaud T., M.A. Beaumont, M. Ciosi, J.M. Cornuet and A. Estoup, 2010. Inferring introduction routes of invasive species using approximate Bayesian computation on microsatellite data. *Heredity*, **104**, 88-99.

Haag-Liautard C., N. Coffey, D. Houle, M. Lynch, B. Charlesworth and P.D. Keightley, 2008. Direct estimation of the mitochondrial DNA mutation rate in Drosophila melanogaster. *Plos Biol*, **6**, e204.

Hamilton, G., M. Stoneking and L. Excoffier, 2005. Molecular analysis reveals tighter social regulation of immigration in patrilocal populations than in matrilocal populations. *Proc. Natl. Acad. Sci. USA*, **102**, 7476-7480.

Hasegawa, M., Kishino, H and Yano, T., 1985. Dating the human-ape splitting by a molecular clock of mitochondrial DNA. *Journal of Molecular Evolution* 22:160-174.

Hudson,R., M. Slatkin and W.P. Maddison, 1992. Estimation of levels of gene flow fom DNA sequence data. *Genetics*, 132, 583-589.

Hudson, R., 2002. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics*, 18: 337-338.

Ihaka R. and R. Gentleman, 1996. *R*: a language for data analysis and graphics. *J. Comput. Graph. Stat.*, **5**, 299-314

Ingvarsson P.K., 2008. Multilocus patterns of nucleotide polymorphism and the demographic history of *Populus tremula. Genetics*, 180: 329-340.

Jukes, TH and Cantor, CR., 1969. Evolution of protein molecules. Pp. 21-123 in H. N. Munro, ed. *Mammalian protein metabolism*. Academic Press, New York.

Kimura, M., 1980. A simple method for estimating evolutionary rate of base substitution through comparative studies of nucleotide sequences. *Journal of Molecular Evolution* 16:111-120.

Lombaert E., T. Guillemaud, J.M. Cornuet, T. Malausa, B. Facon and A. Estoup, 2010. Bridgehead effect in the worldwide invasion of the biocontrol harlequin ladybird. *PLoS ONE*, http://dx.plos.org/10.1371/journal.pone.0009743.

Matsumoto M and T Nishimura, 2000. Dynamic Creation of Pseudorandom Number Generators. *Monte Carlo and Quasi-Monte Carlo Methods 1998*, Springer, pp 56–69.

Miller N, A. Estoup, S. Toepfer, D Bourguet, L. Lapchin, S. Derridj, K.S. Kim, P Reynaud, F. Furlan and T. Guillemaud, 2005. Multiple Transatlantic Introductions of the Western Corn Rootworm. *Science*, **310**, p. 992

Nei M., 1972. Genetic distance between populations. *Am. Nat.* 106:283-292

Nei M., 1987. *Molecular Evolutionary Genetics*. Columbia University Press, New York, 512 pp.

Ohta, T. and Kimura, M., 1973. A model of mutation appropriate to estimate the number of electrophoretically detectable alleles in a finite population.

Pascual, M., M.P. Chapuis, F. Mestres, J. Balanyï¿œ, R.B. Huey, G.W. Gilchrist, L. Serra and A. Estoup, 2007. Introduction history of *Drosophila subobscura* in the New World : a microsatellite based survey using ABC methods. *Mol. Ecol.*, **16**, 3069-3083.

Pollock DD, Bergman A, Feldman MW, Goldstein DB, 1998. Microsatellite behavior with range constraints: parameter estimation and improved distances for use in phylogenetic reconstruction. *Theoretical Population Biology*, **53**, 256?271.

Pritchard, J., M. Seielstad, A. Perez-Lezaun and M. Feldman, 1999. Population growth of human Y chromosomes: a study of Y chromosome microsatellites. *Mol. Biol. Evol.* **16**, 1791-1798.

Pudlo, P., Marin, J-M., Estoup, A., Cornuet, J-M., Gautier, M., and C.P. Robert. 2015. Reliable ABC model choice via random forests. *Bioinformatics*, in review.

Rannala, B., and J. L. Mountain, 1997. Detecting immigration by using multilocus genotypes. *Pro. Nat. Acad. Sci. USA* **94**, 9197-9201.

Raymond M., and F. Rousset, 1995. Genepop (version 1.2), population genetics software for exact tests and ecumenicism. *J. Hered.*, **86**, 248-249

Schug M.D., T.F. Mackay and C.F. Aquadro, 1997. Low mutation rates of microsatellite loci in *Drosophila melanogaster*. *Nat Genet.* **15**, 99-102.

Stephens, M. and P. Donnelly, 2000, Inference in molecular population genetics (with discussion). J. R. Stat. Soc. B **62**, 605-655.

Tajima, F., 1989. Statistical method for testing the neutral mutationhypothesis by DNA polymorphism. *Genetics* 123: 585-595

Tamura, K., and M. Nei., 1993. Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Molecular Biology and Evolution* 10:512-526.

Vï¿œzquez F.J., T. P?rez, J. Albornoz and A. Domínguez, 2000.Estimation of microsatellite mutation rates in *Drosophila melanogaster*. *Genet Res.*, **76**, 323-6.

Weir BS and CC Cockerham , 1984. Estimating F-statistics for the analysis of population structure. *Evolution* **38**: 1358-1370.

Whittaker, J.C., Harbord, R.M., Boxall, N., Mackay, I., Dawson, G. and Sibly, R.M. 2003. Likelihood-based estimation of microsatellite mutation rates, *Genetics*, **164**, 781-787.