

Joaquim F. Pinho Ant



Faculdade de Ciências da Universidade do Porto
Tese de Mestrado em Bioinformática

Tiago Rodrigues Antão

Crossing Informatics and Genes

A skill set in search of a problem

Tese orientada por:
Professor Doutor Gordon Luikart
Professor Doutor Paulo Alexandrino

Acknowledgments

I would like to start by thanking all my Masters' colleagues. We have endured, collectively, the first edition of this degree.

To Maria João Ramos, for being so proactive in trying to sort out problems with the first edition of this degree.

To Miguel Fonseca and Catarina Rato for helping me with some of the work involved in this thesis.

To Maria Joana Ferreira da Silva and Bruno Fonseca Simões for tolerating my bad mood during this long and strange year.

To António Múrias Santos, who was there, teaching a course in the week I was thinking in giving everything up. He gave me hope that things would get on track.

To Paulo Alexandrino, for always believing in me.

Last, but surely not least, Albano and Gordon. The reasons to thank them would be too long to fit an acknowledgments page.

DCC FCUP
BIBLIOTECA
Coloc _____
N.º Reg 1790
Departamento de Ciência de Computadores

Abstract

Bioinformatics is a vast area which is mainly the sum (as opposed to only the intersection) of biochemistry, biology, computer science and statistics. Entering the field is clearly an overwhelming experience, being hard just understanding the many possibilities that exist in terms of research directions. One alternative in deciding which research direction to take is to try several alternatives, getting a practical feel of some of the many problems that remain to be solved and then decide, with some real experience, which direction to take.

The work done in this thesis describes some of the paths probed in the field of bioinformatics to understand the feasibility and interest from the author on some of the many topics available. Two common traits underlie the work presented: all tasks have some form of practical utility (either pedagogical or scientific) and represent *paths not taken*, in the sense that the research direction during the PhD that will follow will not be based on any of the topics presented here.

This work is divided in three parts:

1. BACA, a retriever, organizer and visualizer of multiple mitochondrial (mt) genomes was developed. As retrieving and organizing data from complete mtDNA genomes is a time-consuming task, even more so when comparing several genomes or genes as data retrieval has to be repeated multiple times, we developed an application that takes a GenBank query, retrieves all related genomes and generates multiple FASTA files organized both by genomes and genes. A web-based user interface and an interactive graphical map of all genomes with all genes are also provided.
2. Bairom, a forward-time population genetics simulator was developed allowing for the modeling and analysis of complex scenarios. Concepts like selection, mutation, mating, migration, population structure and demography can be modeled in our framework. The simulator can also be extended with new models of the above concepts. A novel approach is developed to allow for the study of homoplasies. The output includes not only raw data about all populations and generations, but also demographic information, marker analysis, typical statistics (like F_{st}) and information about homoplasies. The system can be used both via an

easy to use web interface or by creating scripts with the libraries provided.

3. Modeler4Simcoal2 (m4s2), an extensible graphical tool to model linked loci and population demographies which can be feed into coalescent simulators was also developed. The application is easy to use, allowing for the modeling of complicated scenarios, making coalescent simulation accessible to a wider range of biologists. The software includes an extension system allowing for new models to be created, published and downloaded from the Internet.

Resumo

A Bioinformática é uma vasta área, sendo essencialmente a soma (em oposição a ser apenas a intersecção) de bioquímica, biologia, informática e estatística. Começar a trabalhar neste campo é uma experiência esmagadora, sendo complicado apenas conseguir entender as muitas possibilidades que existem em termos de direcções de investigação. Uma hipótese no apoio à decisão de qual caminho de investigação que se pode tomar será a de tentar várias alternativas, tendo assim uma sensação real e prática de alguns dos muitos problemas que a área coloca e, assim, decidir com alguma experiência prática, que caminho tomar.

O trabalho feito nesta tese descreve alguns dos caminhos tomados pelo autor no campo da bioinformática, para compreender a exequibilidade e interesse de alguns dos muitos tópicos possíveis. Dois traços comuns atravessam o trabalho apresentado: todas as tarefas têm, de uma forma ou de outra, alguma utilidade prática (pedagógica ou científica) e representam *caminhos que não serão tomados*, no sentido de que a investigação que será feita no doutoramento que se seguirá a este trabalho não será baseada em nenhum dos tópicos aqui apresentados.

O trabalho está dividido em três partes:

1. BACA, uma aplicação para obter, organizar e visualizar múltiplos genomas mitocondriais. Como a obtenção e organização de dados de genomas mitocondriais completos é uma tarefa que consome muito tempo, sobretudo quando é necessário comparar vários genomas ou genes pois a obtenção de dados têm que ser repetida muitas vezes, foi desenvolvida uma aplicação que, dada uma interrogação em formato GenBank, obtém todos os genomas relevantes e gera múltiplos ficheiros FASTA organizados tanto por genomas como por genes. Um interface web que inclui um mapa gráfico interactivo de todos os genomas com todos os genes é também disponibilizado.
2. Bairom, um simulador de genética populacional foi desenvolvido, permitindo a modelação e análise de cenários complexos. Conceitos como selecção, mutação, migração, acasalamento, estrutura populacional e demografia podem ser modelados neste programa. O simulador pode ser extendido como novos modelos dos conceitos acima mencionados. Foi

desenvolvido um novo método para o estudo de homoplasias. O simulador disponibiliza não apenas dados em bruto de todas as populações e gerações, mas também informação demográfica, análise de marcadores, estatísticas típicas (como F_{st}) e informação sobre homoplasias. O sistema pode ser usado tanto através de um interface web fácil de utilizar como através de pequenos programas que podem ser construídos usando as bibliotecas disponibilizadas.

3. Modeler4Simcoal2 (m4s2), uma aplicação gráfica extensível que modela *loci* e demografias populacionais de forma a serem usadas em simuladores coalescentes. O programa é fácil de utilizar, permitindo a modelação de cenários complexos, tornando assim a simulação coalescente acessível a um maior número de biólogos. A aplicação é extensível, permitindo a criação de novos modelos que podem ser disponibilizados e acedidos via Internet.

Contents

Contents	1
List of Figures	3
1 Introduction	5
2 Bairom: An extensible individual-based population genetics forward-time simulator	9
2.1 Introduction	9
2.2 Features	10
2.3 Homoplasies	11
2.4 Web interface	12
2.5 Scenario Modeling	19
2.6 Software architecture	21
2.7 Preliminary discussion	22
3 BACA - Article in press	25
4 BACA: a mitochondrial genome retriever, organizer and visualizer	31
4.1 Architecture and implementation	31
4.2 The GenBank interaction component	32
5 m4s2 - Article submitted	35
6 m4s2	39
6.1 m4s2 components	39
6.2 Supported demographies	39
6.3 Creating new demographies	42
7 Discussion	51
7.1 Comments on the software developed	51
7.2 Sociology of bioinformatics	53
7.3 Making software for bioinformatics	54

7.4 Final comments	59
Bibliography	61

List of Figures

2.1	The relationship among several alleles	12
2.2	The initial page of Bairom's web interface	13
2.3	The Microsatellite page of Bairom's web interface	13
2.4	The "simple" simulation	15
2.5	The first Malaria simulation	16
2.6	The second Malaria simulation	17
2.7	The third Malaria simulation	18
2.8	The Wahlund simulation	19
2.9	The expansion simulation	20
2.10	The microsatellite mutation simulation	20
4.1	BACA software architecture	32
6.1	A decline followed by a population split	45
6.2	A complex demography	48

Introduction

When asking most people about what they think bioinformatics is, the most common answer tends to be a (small) intersection between the areas of computer science and statistics on one side and biology and biochemistry on the other. Most commonly that small intersection is the research direction on which the queried person works. The truth seems to be a bit away from this. In fact, the vast majority of fields of study in biology and biochemistry are amenable to being supported by both computer science and statistics. This essentially means that the majority of fields of the disciplines involved in bioinformatics are “inside” bioinformatics, or, putting it in another way, bioinformatics is, in practice, the reunion of all the disciplines involved. Therefore, entering this area without a clear objective of study, as was the case of the author of this work, is an overwhelming experience as the possibilities are immense. This poses two possible problems: either one spends too much time learning all the alternatives, thus becoming a *specialist in nothing* with no clear research direction, or, in the opposite direction, one takes a decision on the research path too soon, without considering other possibly more motivating alternatives.

A possible path to avoid the scenarios described above is to try to *sample* some of the many research problems available, thus having a *feel* of the ones that a certain individual might feel more inclined and motivated to address on a long term basis. There was a very conscious decision to use the Master’s degree precisely to *sample* and *feel* some of the possible areas for future work to be done. That is, the coherence of the work presented is not based on a scientific path, but on what could be called *soul searching*, i.e., finding, among the many possibilities, what would be the one that might interest the author the most in order to guide a successful long term research path. In this sense, the guidelines of this thesis were not very orthodox, but they had, in fact, quite a favorable outcome: not only some content has been already accepted for publication, but also, more importantly, it allowed the author to discover a

personally highly motivating research path (while developing software to help collaborators more easily analyze their genomic data sets).

What were then, the criteria to develop the *sampling* work? First of all, answering a real question or need posed by any of the researchers on the life sciences “side” (in the sense that the author has a computer science background and was interested in understanding what were the main problems entertaining life scientists). Secondly conducting a task from the beginning to the absolute end, that is assuring that all the issues necessary to completing the task would be handled successfully, this to have, in as much as possible, a complete view of what was involved on each task.

The three pieces of work which are presented here, of the several tried, were chosen for two reasons: they had some measure of success (either because they have some pedagogical or scientific value. The scientific value is accessed through the fact that part of the work (one manuscript) was already published[2] on a peer reviewed journal and another part is currently in submission); they were also chosen because they are not what will be directly pursued by the author on his future (PhD) work, although part of them are strongly related.

The three parts are presented chronologically by when they were developed. In reality the first and the third have some measure of relation, in the sense that they are both related to simulation in population genetics.

Bairom: A forward-time population genetics simulator Originally, an individual based forward-time population genetics simulator was developed allowing for the modeling and analysis of complex scenarios. Concepts like selection, mutation, mating, migration, population structure and demography can be modeled in our framework. The simulator can also be extended with new models of the above concepts. A novel approach is developed to allow for the study of homoplasies. The output includes not only raw data about all populations and generations, but also demographic information, marker analysis, typical statistics (like F_{st}) and information about homoplasies.

The system can be used both via an easy to use web interface or by creating scripts with the libraries provided. Although the author’s main line of research is currently population genetics, this simulator was done even before being exposed to population genetics concepts, there is some naïvety in the design and implementation of the software. The text presented was designed with the sole purpose of being understood by biologists with little or no knowledge of computer science. In that sense it was an interesting exercise in *cultural adaptation*. As such discussion of more technical programming and extensibility issues are kept to a bare minimum, for instance the fact that the system is completely extensible because it can be linked as a programming library is not addressed at all. Furthermore there were much care put on creating and documenting a

easy to use web interface. The cultural adaptation and communication issue that is addressed should not be underestimated as it continuously shows up as a problem in multidisciplinary teams as the one the author is currently involved with. There is a conscious decision of not trying to publish this simulator, which is explained in the discussion chapter.

BACA: A mtDNA genome analysis software A retriever, organizer and visualiser of multiple mitochondrial genomes was then developed. As retrieving and organizing data from complete genomes by hand is a time-consuming task, even more so when comparing several genomes or genes as data retrieval has to be repeated multiple times we developed an application that takes a GenBank query, retrieves all related genomes and generates multiple fasta files organized both by genomes and genes. A web-based user interface and an interactive graphical map of all genomes with all genes are also provided. This work was initially developed to help study mitochondrial gene rearrangements and partial genome duplications [9]. It can also be used to support most work that requires retrieving, organizing or visualizing mitochondrial genomic data. This was developed in a typical setting where a very clear automation need posed by life scientists was addressed by a computer scientist as subsidiary, but nonetheless important, work. This work was published on *Molecular Ecology Notes* and the article can be found in chapter 3.

Modeler4Simcoal2 (m4s2): A modeler for simulating the coalescent

An extensible graphical tool to model linked loci and population demographies which can be feed into coalescent simulators was also developed. The application is easy to use, allowing for the modeling of complicated scenarios, making coalescent simulation accessible to a wider range of biologists. The software includes an extension system allowing for new models to be created, published and downloaded from the Internet. This tool, although using Simcoal2 [14], could easily be adapted to model demographies and linked loci on the developed forward-time simulator. Of notice is the fact that this program is a *spin-off* of the long term work being conducted on evaluating the robustness of methods for detecting selection and molecular adaptation [4] and developing new methods especially ones based on Approximate Bayesian computations (ABC) [24]. The module used to generate simulations with varying nuisance parameters was repackaged inside a programmable user-interface. This work is currently being submitted to peer reviewed journals, the submitted text to the journal *Bioinformatics* can be seen on chapter 5.

This Masters thesis includes one chapter on the population genetics simulator, one published article and one chapter with extra information on the mtDNA analysis tool and one submitted article with one chapter on the modeler for simulating the coalescent. A final discussion closes this text.

Bairom: An extensible individual-based population genetics forward-time simulator

2.1 Introduction

Molecular markers like Single Nucleotide Polymorphisms (SNPs) and Microsatellites are fundamental to gain insights regarding the evolution of natural populations. There are quite a few methods [17][4] to estimate important population genetic parameters, but either these methods impose restrictions on the populations being studied or their behavior is simply unknown under new conditions. It is fundamental to know how these methods perform under new situations and if they are resilient to changes in their assumptions about the populations under study. One possible approach is to simulate in a computer important population models and then apply the existing analysis methods to check if they detect the relevant parameters.

The most common approach is to simulate using coalescent based methods [21][15], another possibility is to do forward-time simulation [3][18]. With forward-time simulation it is possible to keep track of the full ancestral generation information and have more flexibility in modeling population genetics concepts, adding to that, coalescent methods are mainly developed for haploid individuals. Although forward-time simulations are more computationally demanding, current personal computer technology makes them feasible.

We developed Bairom, a new population genetics simulator. Bairom implements the most fundamental population genetics concepts such as selection, mutation, mating, migration, population structure and demography. It sup-

plies the most common operators for these concepts and also allows for user creation of new ones. A novel approach was created to allow the study of homoplasies. The program outputs raw data of all populations and generations simulated, it also computes some typical population genetic statistics (e.g., F_{st} and expected heterozygosity). The simulator can be used inside a easy to use web interface.

In the first section we discuss all the implemented features with the exception of the ones related to homoplasies. The second section is dedicated to homoplasies. The third presents the web interface discussing several typical population genetics examples from a user point of view, it is followed by a discussion using the same examples but from a modeling and programming point of view. Then we present the software architecture. We do some considerations about testing and assuring the reliability of the program. The final section is a discussion of this work and possible future development directions.

2.2 Features

Concepts

Fundamental to any individual-based simulation is the notion of individual. In our case, an individual has a gender, a set of “mitochondrial” loci and two sets of nuclear loci. Any of the sets can be empty, thus allowing simulation of only haploidy or diploidy if desired. “Mitochondrial” alleles can simulate not only maternal but also paternal (“Y-chromosome”) genetic transmission.

A gene is not a sequence of DNA but a set of possible SNPs (generated as new mutations arise) and a set of microsatellites near it, this notion of gene is much more pragmatic in a animal population genetics environment than the typical model of a gene as a sequence of nucleotides as it models the concepts that animal population geneticists mostly use in their daily work.

It is important to note that recombination among genes is not supported as we have no notion of distance among them. But, recombination is supported inside a gene, as microsatellites and SNPs have positions relative to the gene.

Mutation on both Microsatellites and SNPs is available. For Microsatellites the Stepwise Mutation Model is offered. The user can create new mutation models.

The following demographic models are available: constant size, immediate shrinkage or expansion, linear shrinkage or expansion over several generations and the application of sequence of different models (allowing to specify, for instance constant size followed by an immediate shrinkage followed by linear expansion). Again, the user can create new models if necessary. It is important to note that demography is not orthogonal with selection, as an example if we have constant size with a Malaria selection operator that removes 5% of individuals then we have really 5% shrinkage per generation.

Selection can be implemented either by mating preferences (discussed in the next paragraph) or by removing individuals with certain traits. We provide a “Malaria” selection operator which removes a percentage of individuals that are homozygotic for a certain allele of a certain gene. Selection is typically a case where it makes more sense for the user to develop new operators (as they can vary a lot) than for the system programmer to try to offer a set of cast-in-stone operators.

Random mating is provided, as is a function where a percentage of females prefer heterozygotic males for a certain gene (thus indirectly causing balancing selection[17]). Again, the user can extend the system if desired.

It is possible to migrate any quantity or percentage of individuals from one population to another.

Population structure is offered via migration events that repeat every generation. The Island Model and the Stepping-Stone Model (uni- and bi-dimensional) are implemented. Once more, the user is free to extend the system with any other model.

Analysis

For each population, basic demographic information is produced, namely the number of individuals and how many females exist in each generations.

For the whole simulation and for each generation and gene the basic statistics are calculated, namely F_{ST} , expected and observed heterozygosity.

For each gene it is also provided a summary of allele (and pseudo-allele) counts.

Data export

The supplied analysis tools are meant mainly to support a preliminary analysis of the data and for educational purposes.

The data is made available in raw format to allow the user to analyse it in a completely independent way. A utility to load data in R[25] is supplied.

A fundamental issue is to allow analysis programs to study the data produced by Bairom, as such, an exporter is provided for `fdist2`[4].

2.3 Homoplasies

Microsatellites are subject to size homoplasy[7]. Size homoplasy might be problematic when doing populations genetics' analysis if mutation rates and populations sizes are both high.

Bairom allows for the study of homoplasies by recording, for each new allele occurring from a new mutation, both the originating allele and, if it exists the allele for which the new one is homoplastic. Its is also possible to know, for each homoplasy, when it appeared and how long it lasted (in case

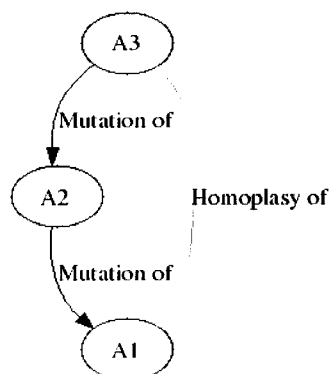


Figure 2.1: The relationship among several alleles

it was subject to genetic drift). As an example, if there is an allele A1 of a certain gene which had 10 repeats on a specific microsatellite and there is a new mutation A2 with 11 repeats and another mutation A3, from A2 with again 10 repeats, the system would record that A3 was originated from A2, A2 from A1 and that A3 is an homoplasy of A1. This is graphically depicted on figure 2.1. We will call A3 a pseudo-allele of A1.

With Bairom it is possible to answer questions like these:

- For a certain number of repetitions of a microsatellite, how many homoplasies are involved?
- Which pseudo-allele it the oldest?
- Which pseudo-allele is the most represented?
- The average age of all pseudo-alleles?

The current analysis tool makes use of some of the information available to plot a graphic of pseudo-alleles occurring during the simulation.

2.4 Web interface

Regarding the input, its is not possible to create a web interface that offers all the expressive power of Bairom or that interface would be so complicated that it would be quite difficult to use. As such the option was to implement a set of simple interfaces that model typical cases used in biology, part of the initial page of Bairom's web interface can be seen on figure 2.2. Each scenario allows for the setting of parameters relevant to it, as an example the interface of the Microsatellite scenario is depicted on figure 2.3.

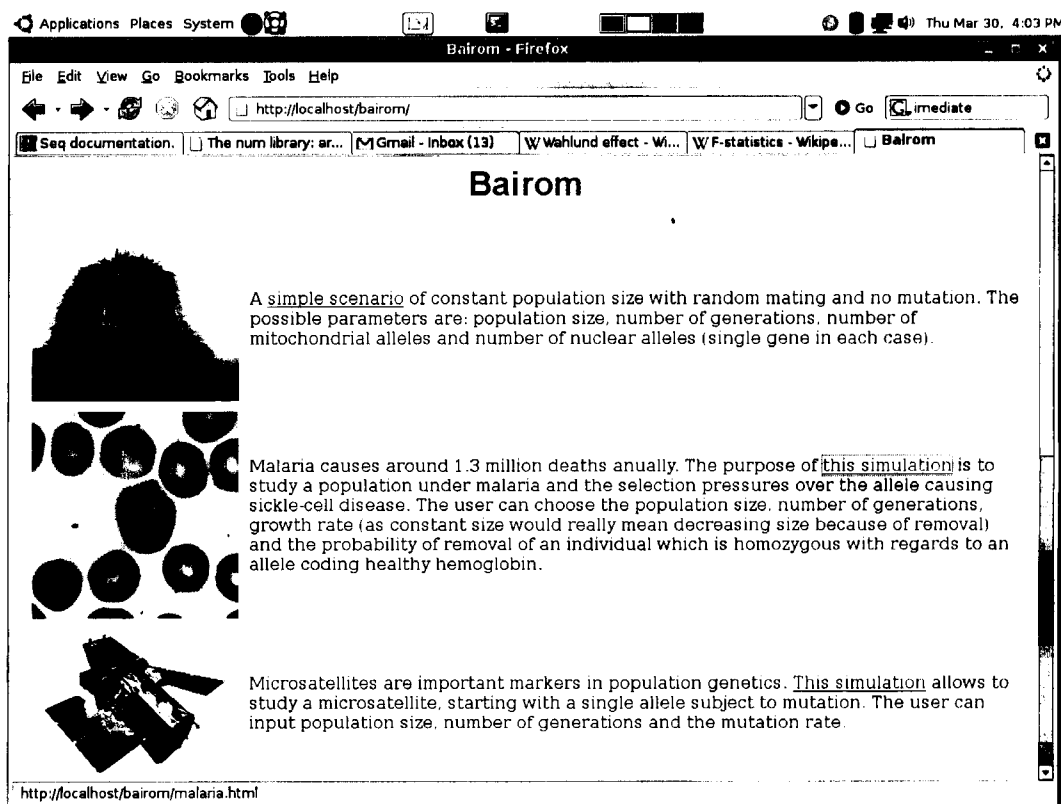



Figure 2.2: The initial page of BairoM's web interface

BairoM - Microsatellites



Initial Population size

Number of generations

Mutation rate

Note:
 1000 - probability 100%
 1 - probability 0.1%

Figure 2.3: The Microsatellite page of BairoM's web interface

For all scenarios the format of the output is the same. Not only a typical web output with per generation and population graphics and statistics is supplied but also the raw data and R scripts with data within are made available in the web application.

Scenarios supported

The following scenarios are supported with a single population:

- **Simple scenario:** Constant size, random mating, no mutation. The parameters are: the initial (and constant throughout time) population size, the number of alleles of the single mitochondrial and single nuclear genes.
- **Immediate contraction or expansion:** Immediate contraction or expansion, random mating, no mutation. The extra parameters compared to the simple scenario are: time for the event to occur and percentage of change of the population size. It is not possible to specify the number of alleles of any gene.
- **Spanning-generation contraction or expansion:** Contraction or expansion spanning several generations, random mating, no mutation. The extra parameters compared to the initial scenario are: time for the event to start, time for the event to end and percentage of change of the population size in each generation the event lasts. It is not possible to specify the number of alleles of any gene.
- **Malaria:** This scenario removes (selects) a part of the individuals which are heterozygotic for a certain bi-allelic “blood” gene. It disallows individuals that are homozygotic for a certain allele to be born. The user can specify the removal probability and the growth rate of the population, plus initial size and number of generations
- **Microsatellite mutation:** This scenario mutates a gene with a single microsatellite. The user can specify the mutation rate along with population size and number of generations.

The following scenario is supported with multiple populations:

- **Wahlund:** In the initial generation, two populations exist and then they are merged at some point in time to create a single population. The user can specify the initial size of each population, the number of generations the simulation will run, the number of alleles of the nuclear gene and the generation where the populations will be merged.

We now exemplify the usage of the scenarios available on the web interface.

Simple scenario

We start by showing, on figure 2.4, the simple scenario simulation, having 100 individuals, 50 generations, 5 mitochondrial alleles and 6 nuclear alleles.

Obviously the simulation is non-deterministic and, as such, the results vary from execution to execution.

In this case we see a typical pattern of allele evolution in a small population with genetic drift happening both on the mitochondrial and nuclear gene.

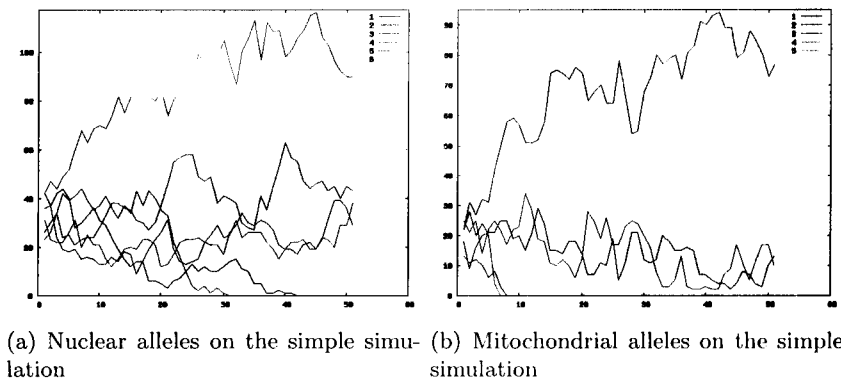


Figure 2.4: The “simple” simulation

Malaria

With regards to Malaria, we will present three scenarios, being two of them in more extreme conditions.

Our first scenario with Malaria (figure 2.5) includes 200 individuals, 50 generations, 35% removal rate of homozygous individuals and a growth rate of 21%.

First, as we can see in the demography (figure 2.5(a)), there is no growth of 21%, on the contrary there is shrinkage. That happens because the rate of removal caused by the disease is bigger than the population growth rate.

We can also see that the expected heterozygosity is smaller than the real one (figure 2.5(b)), as a certain homozygotic configuration is not allowed and there is positive selection for the heterozygotic one via Malaria.

As expected there are more normal alleles than sickle-cell ones (figure 2.5(c)).

One first extreme scenario (figure 2.6) is one population where the removal rate is 0 (no Malaria). We will consider a 2% growth rate.

In this case, there is in fact growth (figure 2.6(a)). Although the rate is much smaller than in the previous example as there is, in this case, no removal growth really happens.

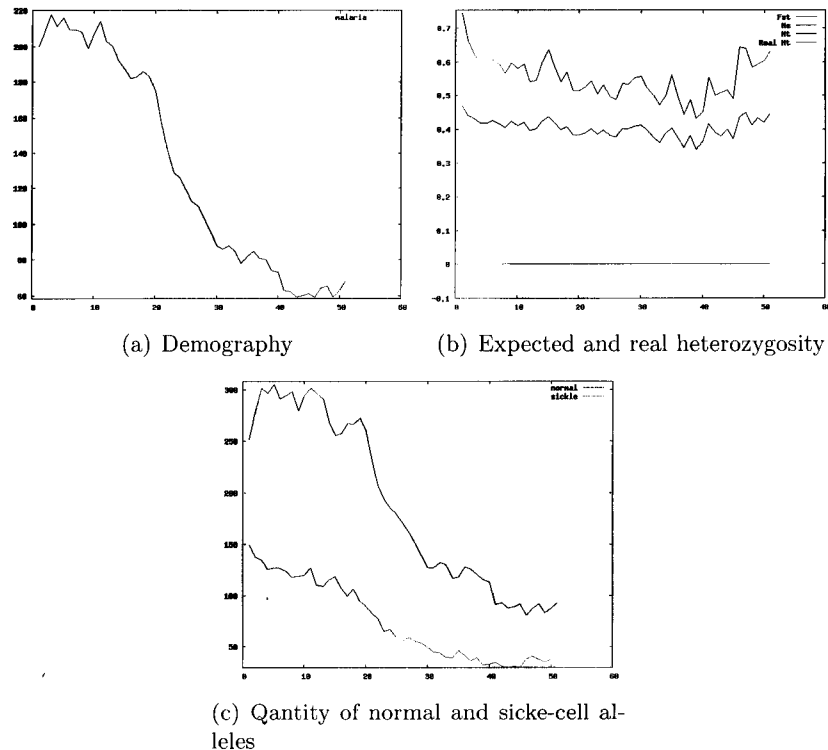


Figure 2.5: The first Malaria simulation

We can see the evolution of both expected and real heterozygosity in figure 2.6(b) and that both fall sharply as there is strong pressure to remove the sickle-cell allele (sickle-cell homozygotes are not viable). In fact, in such a small population the sickle-cell allele disappears (figure 2.6(c)).

Another extreme case (figure 2.7) would be to have a removal rate of 95%. In this case we run 100 generations and a growth of 50%.

As the initial generated population has a big quantity of normal alleles we have an initial fall in the demography (figure 2.7(a)), which will change direction when there is a rise in heterozygosity (which protects from the disease) as can be seen on figures 2.7(b) and 2.7(c).

Wahlund

In this simulation (figure 2.8) we create 2 populations (with a single nuclear gene with a user-specified number of alleles), each with 30 individuals, we run the simulation for 100 generations and we merge the 2 populations on generation 50.

As we can see on figure 2.8(a), the 2 initial populations have 30 individuals until generation 50 where they merge to form a new population with 60

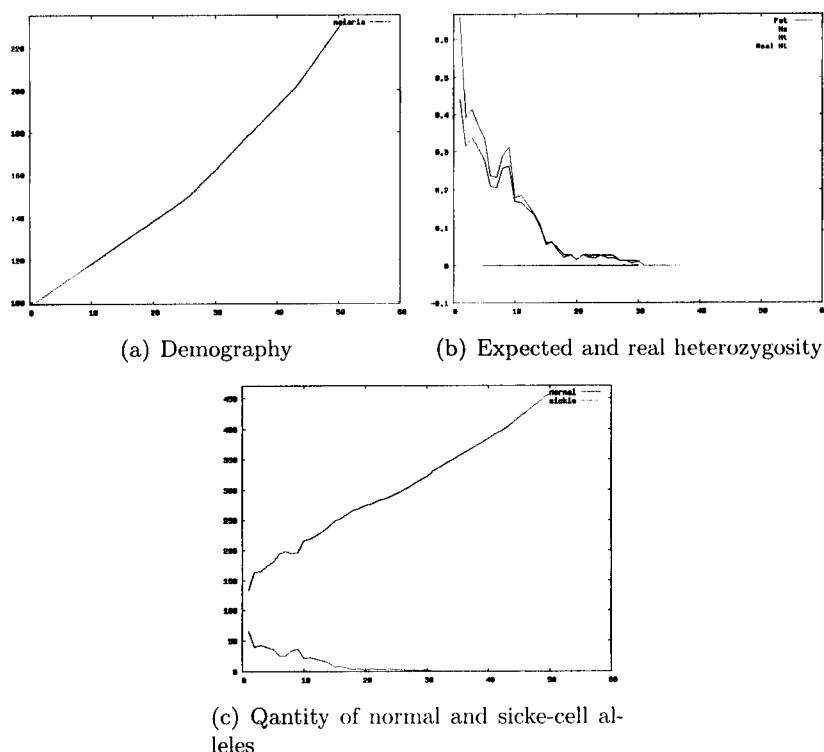


Figure 2.6: The second Malaria simulation

individuals.

As we can see on figures 2.8(b) and 2.8(c), the alleles that get fixated on both populations are different, this causes F_{ST} to grow as can be seen on figure 2.8(d). On generation 50 F_{ST} abruptly falls to 0 as we are now dealing with one single population.

Contraction/Expansion

It is both possible to have immediate growth/shrinkage, in a single generation and growth/shrinkage spanning several generations. As the immediate growth scenario is more common we will only run the multiple-generation one (figure 2.9).

We will start with 100 individuals and have 10% growth from generation 15 to 25, ending the simulation on generation 50. Regarding demography the simulation behaves as expected (figure 2.9(a)), it is interesting that we see genetic drift after the growth (mainly because the population is still small - figure 2.9(b)).

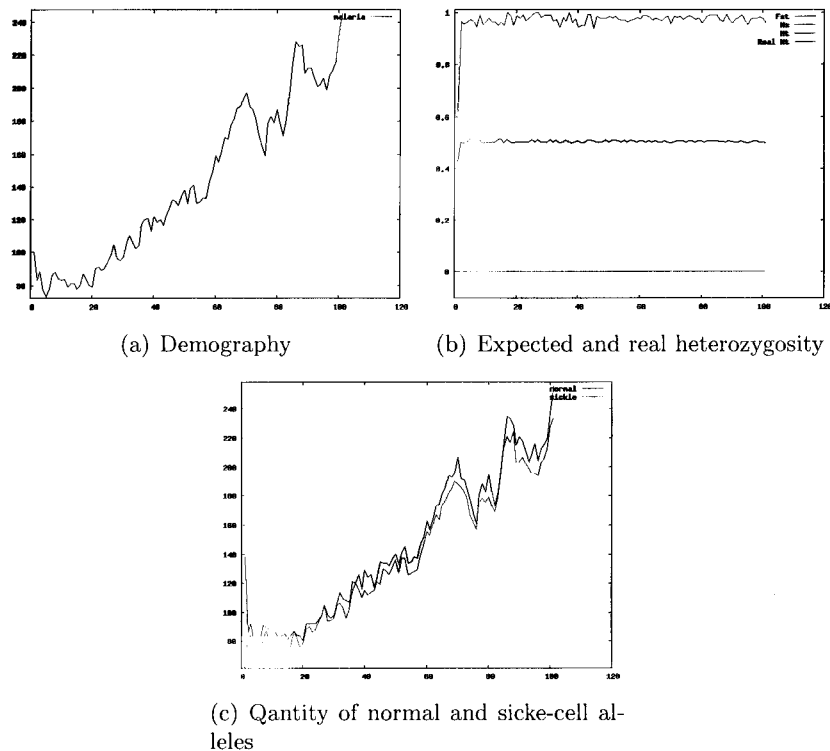


Figure 2.7: The third Malaria simulation

Microsatellite mutation

In this scenario (figure 2.10) we will concentrate only on comparing alleles with pseudo-alleles. We will have a population of 1000, 100 generations and a 1% mutation rate. The reason for such a mutation rate is that, with such a small population and no growth, the overwhelming majority of mutations will just drift. It is probably much more interesting to do this scenario with strong population growth and much more individuals. This is a first intuitive indication that homoplasies could only be really interesting with large populations subject to strong population growth.

As we can see from both graphics 2.10(a) and 2.10(b), the story can be quite different comparing real and pseudo alleles. Furthermore, whereas we have 6 real alleles, we had, during the simulation run around 600 pseudo-alleles. There is no pretension of thinking that this simulation simulates any natural process (especially with the 1% mutation rate). To understand if this simulation mechanism is in anyway useful requires further study.

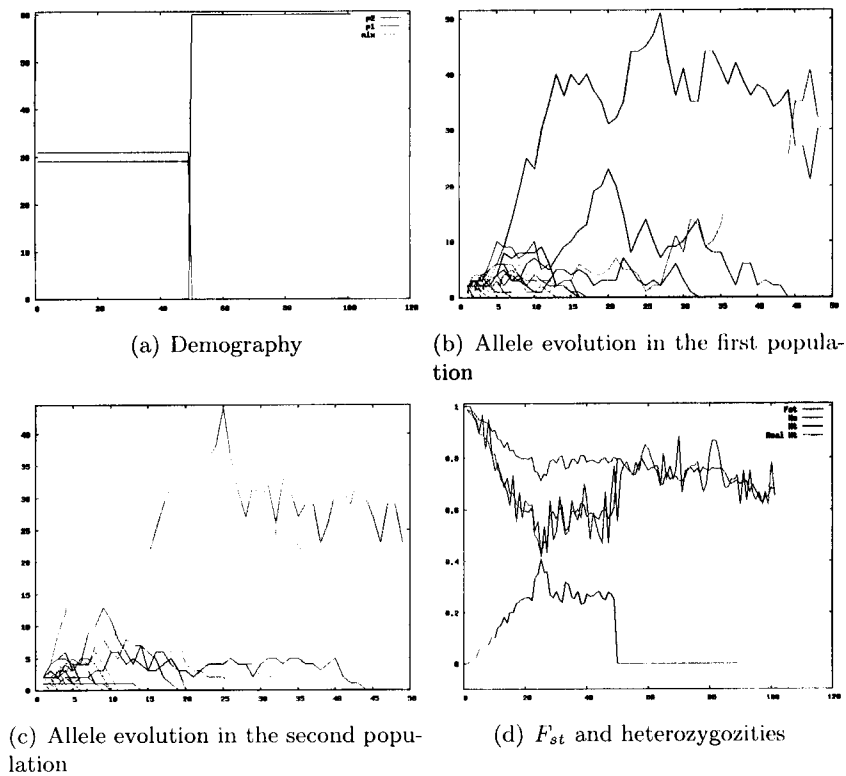


Figure 2.8: The Wahlund simulation

2.5 Scenario Modeling

We now describe some fundamental issues in modeling some of the previous scenarios. It is important to note that Bairom allows for the modeling of much more scenarios. These ones are described only because there is a web interface for them.

A decision was made of not including any code examples while discussing the modeling and programming issues.

Simple

The simple scenario is, from a modeling point of view... simple. A species is created with two genes, one mitochondrial and other nuclear, each gene has the number of alleles specified by the user. A population is created with a user-specified number of individuals

The simulation is then run for the requested number of generations under random mating and constant size. There is no mutation.

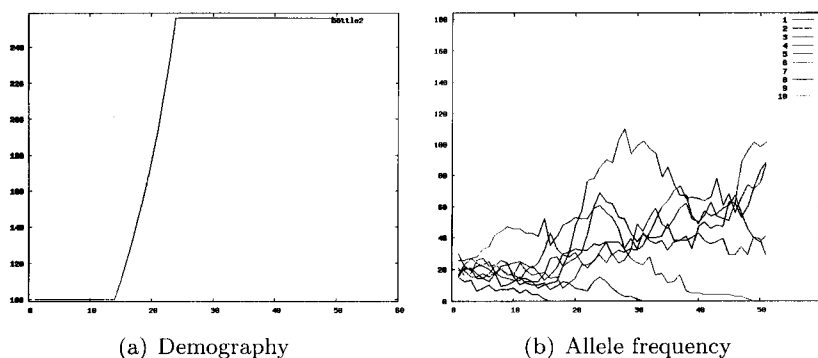


Figure 2.9: The expansion simulation

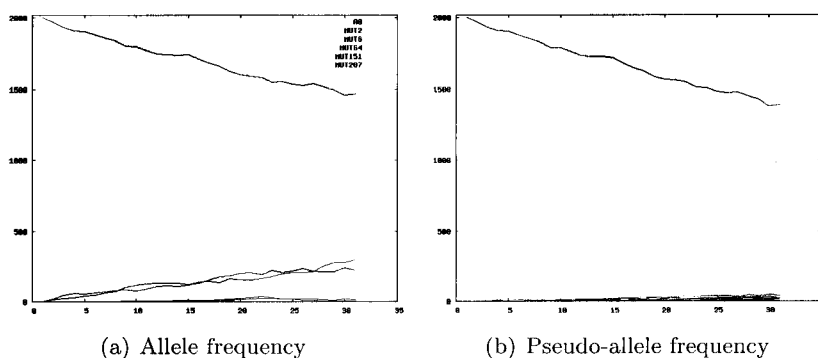


Figure 2.10: The microsatellite mutation simulation

Contraction/expansion

The immediate contraction/expansion is quite simple, the constant size function of the simple scenario is changed by a immediate bottleneck function which is constant size for all generations except a specified one where growth/shrinkage occurs.

The generation-spanning contraction/expansion requires the application of three growth functions: constant size followed by linear growth/shrinkage for a certain quantity of generations followed again by constant size.

Microsatellite mutation

Although the Microsatellite scenario involves mutation, its modeling is quite simple: A nuclear gene with a single allele including a microsatellite is created. The simulation is then run with the built-in SMM operator.

Wahlund

The Wahlund scenario requires the creation of two populations with user defined size and of a migration event to merge the two populations in a new one.

Malaria

The Malaria scenario is the most complex one to model, three steps are needed:

- The creation of a "blood" gene, with 2 alleles. One "normal" allele and one "sickle-cell" allele. There is no need to include markers on the gene.
- A selection operator which removes homozygotic "normal" allele individuals with a certain probability.
- An offspring validation function which disallows homozygotic "sickle-cell" allele individuals.

2.6 Software architecture

Bairom is made of 5 components:

- The fundamental one is the simulator itself. The simulator simply generates raw data (information about individuals in populations and generations and information about the species - like new alleles created during the simulation run) according to a script supplied by the user (or predefined scripts used in the web interface). It was developed in OCaml.
- The second component is the analyser, which mainly generates all the analysis about demography, allele distribution, F_{ST} , ... The main use of the analyser is to generate data to the web interface. It was developed in OCaml.
- The most visible and complex part is the web interface¹, which is the part taking the user input from the web and generating the output in HTML format for the web. That is to say, that for strict research purposes the most complex part of the system is not very important. Its was developed in Perl.
- A fundamental, but, nonetheless quite simple, component for research purposes is the data exporter to fdist2 (and other formats in the future).

¹In fact the complex parts of the web interface are coded in the analyser. Without spending too much time in software design considerations, it should be said that most of the complexity in creating this tool is due to the web interface.

It is the exporter features that make Bairom useful in a research context where more proper analysis tools exist other than the internal analyser. It was developed in OCaml.

- An exporter that can be seen as a separate component is the one based on R. R, in opposition to `fdist`, `genepop` or `arlequin` can be used for general purpose analysis of the data. It was developed in R.

2.7 Preliminary discussion

Bairom can be used both as a research and educational tool. This document and the current implementation are clearly more geared towards an educational approach: Some effort was put into creating a easy to use web interface outputting simple analysis as a result. Furthermore, from a research perspective the most interesting use will be running the software hundreds or thousands of times with the same scenario and calculating statistics over the results from the these simulations, that is, easy to use user interfaces will play a very small role. It might be seen as unintuitive to the biologist, but, from a software development point of view it is much harder (and less interesting) to develop an easy to use GUI or web interface than it is to develop a research infrastructure. The expectations from biologists about GUIs and web interfaces are, most probably, exaggerated. Further comments about this issue are available on the final discussion chapter.

A future use of this tool as a research vehicle would at least require:

- Studying the problem of initial allele distribution. The current initial distribution of alleles is done with equal probability to all alleles. This hardly mimics natural populations. A way to mitigate this problem is to have so many alleles, compared to the initial population size, that in the first generations there will be massive drift. But, for most natural scenarios this is still quite useless.
- As explained before, the typical use-case for research purposes will be to run a massive number of times the same scenario and collect statistics from multiple runs. Although there are already exporters for data to other programs like R and `fdist2`, some more work on handling data from multiple runs of a simulation should be done.
- For the same reason, i.e., running a massive number of simulations a parallel version could be done.
- The support for hierarchical population structure is still quite poor.
- More migration models could be included.
- More mutation models could be included.

- Although it is possible to study homoplasies with the software, no study was done. The biggest usefulness of the program for this purpose was not assessed in practical terms.
- N_e . With random mating the effective population size is, in fact, the population size. But, in quite a few cases other types of mating are important. The tool should be able to point N_e . It should be noted however, that with forward-time individual simulations it is not possible to have tight control over N_e , as lots of operators influence it.
- One of the most fundamental issues is to redesign the programming interface in such a way that it is much easier to use for a biologist. Although a primary design goal of tool was to be easy to program by a non-programmer, it is clearly possible to do much better and to restructure the tool in a more elegant and intuitive way.

Although the number of issues needed to be tackled to use this tool for research is still quite high, they are much easier to approach (with the exception of initial allele distribution) than their counterparts on GUI development. For example including the simulation support for hierarchical populations is about a couple of days work, the same support in the web interface would probably take a couple of weeks. Another example are mutation models: it is absolutely trivial to implement more models.

A fundamental issue which was not tackled was the validation of the program. Without that step (which has to be done if the tool is to be used in the future) the data output will never be trustable, even if, intuitively it seems to generate good results.

Limitations against other simulators

There are a few issues in which Bairom might compare unfavorably against other population genetics simulators:

There was the conscious decision of not allowing overlapping generations. It would make the software much less simple to understand and it was not clear that there are many use-cases that justify the added complexity of generation overlap.

Recombination can only be done inside a gene (as a gene is also a set of microsatellites near it). Namely it is not possible to have recombination between genes as there is no notion of chromosome. This is seen as a possible drawback in the current genomic scenarios and will most probably be addressed in future versions

Although SNPs were not discussed in this text, the support for SNP mutation is fully included in the tool.

BACA - Article in press

In this chapter the BACA article[2], published on *Molecular Ecology Notes* can be found. Currently the article is available as *Online Early*.

PROGRAM NOTE

BACA: a mitochondrial genome retriever, organizer and visualizer

T. ANTAO, A. BEJA-PEREIRA, M. M. FONSECA and D. J. HARRIS

*CIBIO—Research Center in Biodiversity and Genetic Resources, Universidade do Porto, Campus Agrario de Vairao, R. Padre Armando Quintas, 4485-661 Vairao, Portugal***Abstract**

Retrieving and organizing data from complete genomes is a time-consuming task, even more so if the interest lies only in part of the genome (for nongenomic analysis). Furthermore, when comparing several genomes or genes, data retrieval has to be repeated multiple times. We present *BACA*, a software for retrieving, organizing and visualizing multiple mitochondrial genomes. *BACA* takes a GenBank query, retrieves all related genomes and generates multiple FASTA files organized both by genomes and genes. A web-based user interface and an interactive graphical map of all genomes with all genes are also provided. The program is available from <http://cibio.up.pt/software/baca>.

Keywords: barcoding, data retrieval, GenBank, mtDNA, Perl, visualization

Received 30 July 2006; revision accepted 6 November 2006

With the increasing availability of complete genome sequences on public databases it is becoming feasible to do both genome-wide analysis and genome-wide comparisons involving multiple species (Hassanin *et al.* 2005). Indeed the number of complete mitochondrial genomes sequenced for multiple species is growing at a very fast pace. Retrieving and organizing data from several complete genomes is time consuming and takes valuable research time. The availability of entire genomes (i.e. mitochondrial) will open an important door to the study of evolutionary processes, speciation and species identification (Boore 1999). For example, the 'barcoding of life project' will greatly profit from a computational tool that permits automatic retrieval of multiple complete mitochondrial genomes from GenBank. Also, applied molecular ecology studies can profit from this tool, using it to find mitochondrial genes that show peculiar characteristics in a species compared with closely related taxa. This can be particularly useful to develop markers for identification of species or a group of species. To solve this we constructed a software application named *BACA*, with the objective of automating the downloading, organizing and presentation of data from mitochondrial

genomes from multiple species. *BACA* takes a simple GenBank query, retrieves all the related genomes, analyses them and outputs FASTA files organized by genome (one FASTA file per genome, including all the genes of that species) and by gene (one FASTA file per gene, with that gene from all genomes where it exists). Transfer RNAs (tRNA), ribosomal RNAs (rRNA), stem loops, D-loops and the origin of replications can also be analysed. Statistical information is generated for each gene and genome, and scripts are made available to load this in R (R Development Core Team 2005). The interface with the user is performed via a very simple web front end. The program can also visualize all the genomes graphically, comparing genome sizes and gene positions.

Other applications or databases can perform similar functions. Compared to GoBase (Korab-Laskowska *et al.* 1998), *BACA* is more focused in scope, whereas GoBase is a database of organelles (including information on DNA, protein expression, tertiary structures among others) feeding on several existing databases. The focus of *BACA* is mitochondrial DNA, consulting GenBank database on every query and organizing data for the purpose of genetic and genomic comparisons across different taxa. Visualization is also more targeted towards helping in comparative analysis. Compared to the Polymorphix (Bazin *et al.* 2005) database,

Correspondence: Tiago Antao, Fax: +351 252661780; E-mail: tiagoantao@gmail.com

2 PROGRAM NOTE

again BACA is more specifically focused on mitochondrial information. Visualization in BACA targets comparing genomic structures whereas Polymorphix emphasizes more genetic and phylogenetic analysis. Therefore, our application, although allowing single gene analysis like most other applications, can also support comparative genomic studies. As the scope of BACA is more focused than most applications, it is comparatively easier to use and more efficient in supporting mitochondrial multigenomic analysis (with both data retrieval and visualization targeted and optimized to its expected usage patterns) at the expense of a broader scope of usage.

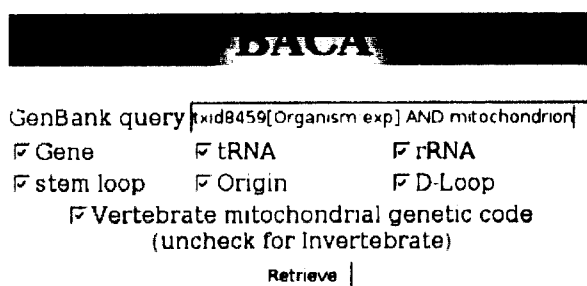
The software was developed using BioPerl (Stajich *et al.* 2002) and the scalable vector graphics (SVG) Perl module. A web server that can run Perl CGI scripts is required (i.e. Apache or Microsoft IIS). Any operating system where Perl is available can be used.

For the user interface, a browser with SVG support is recommended. Full functionality is available without SVG support, although the experience is less interactive.

The program was developed in an extremely applied environment, with the purpose of responding to very clear needs. As such its interface and inner workings are geared towards solving, in a fast, pragmatic and easy way the problem of retrieving, organizing and visualizing genomic mitochondrial data. As BACA was designed to address very concrete needs, some of the statistical information generated is very specific (such as counts and skews related to fourfold redundant sites in protein-coding genes) and aimed towards analyses of base composition variation across taxa (e.g. Boore 1999; Hassanin *et al.* 2005; Xia 2005; Fonseca *et al.* 2006) and (Rest *et al.* 2003). It is important to note that the specificity of some of the information generated does not preclude the use of the program in more general scenarios.

The typical program usage pattern is as follows:

- 1 The user accesses GenBank web interface, searches for complete genomes of interest and finds a query that retrieves all the relevant genomes (for instance a query for mammals retrieving only mitochondrial genomes).
- 2 The user puts the query on BACA.
- 3 BACA retrieves and organizes the data creating FASTA and GenBank files, statistical information and a visualization interface.
- 4 Mitochondrial genomes visualization: the current interface allows the user to easily spot differences in size among genomes, gene positions and existence of annotation for certain features like the origin of replication.
- 5 Using the organized data, for example, by aligning the generated gene FASTA files.
- 6 The user analyses the generated statistics regarding skews in fourfold redundant sites either through the web interface generated or by loading statistical information in R.



The screenshot shows the BACA web interface. At the top, there is a black header with the word 'BACA' in white. Below the header, there is a text input field containing the query 'txid8459[Organism exp] AND mitochondrion'. To the right of the input field is a 'Retrieve' button. Below the input field, there are several checkboxes for selecting features: 'Gene', 'tRNA', 'rRNA', 'stem loop', 'Origin', 'D-Loop', and 'Vertebrate mitochondrial genetic code (uncheck for Invertebrate)'. The 'Vertebrate mitochondrial genetic code' checkbox is checked.

Fig. 1 BACA query interface. This query retrieves all placental mammals mitochondrial genomes including all the supported annotated features.

The software presents a simple interface (Fig. 1) where a GenBank query is given as the input along with the desired features to be organized in FASTA files. As data retrieval can take some time, the response from the system is not the data files but a job number that can be used to retrieve the data in the future. The speed of the operation depends mainly on the number of genomes involved, the network speed and the server load at GenBank. BACA is not computationally intensive.

The output is not only the FASTA formatted data files organized by genome and gene (or other requested features) but also the full GenBank entries and extracted metadata, such as gene positions. For convenience, all the output files are zipped in a single file.

Example usage: We will retrieve, organize and visualize all the mitochondrial genomes for placentals, generating FASTA files for all the features. The query for this operation is: txid9347(Organism:exp) and mitochondrion. The first part of the query can easily be found by using National Center for Biotechnology Information's (NCBI) TaxBrowser; the second part assures that we only retrieve mitochondrial genomes as there might be also nuclear genomes in the database for a certain class. In the case of placentals there is at least information for the nuclear genome of *Homo sapiens*. The input for the web interface can be seen on Fig. 1.

The result, concerning the retrieval and organization part, includes a zip archive with several files inside it. There will be a pair of files per feature, one being a FASTA file including that feature from all species that have it (this file can, afterwards, be aligned by another tool such as CLUSTAL W; (Chenna *et al.* 2003) and another with metadata such as position in the genome and applicable statistics to that feature, for example information on fourfold degenerate sites in the case of protein-coding genes. The same kind of information will be available, organized also by genome. All the retrieved GenBank files are made available. The names of the files inside the zip simply identify what is

Selected object - Home Zip with all files - R file
 Species Homo sapiens Id NC_001807
 Type GENE Description ND5

Mouse over object
 Species Rattus norvegicus Id NC_001665
 Type OENE Description ND5

Visibility: cDNA tRNA rRNA D-Loop Light origin



Fig. 2 Screenshot of the genome visualizer. Both genes and tRNAs are selected.

inside, for example, GENE_gene_ATP6.FASTA has all ATP6 genes from all genomes that have that gene. This is important as the number of generated files can be quite substantial. Regarding the visualization part, a straightforward console is created (Fig. 2).

Most of the console (bottom part) is occupied by the mitochondrial map explorer, where all mitochondria can be found along with the features that the user selected to display. On the top right side there is an information frame whose content changes when the user moves the mouse around, showing the species, GenBank ID, type of feature (e.g. protein-coding gene) and name of feature (e.g. CYTB) below the mouse. If the user clicks on a feature then it becomes available on the top-right frame of the console. On that part of the console the user can also choose which features are visible (protein-coding genes, tRNAs, rRNAs, D-Loops, stem loops and origins of replication). If the user clicks on any of the elements of the top-right frame then explorer on the bottom part is replaced by relevant information regarding the chosen object (e.g. all available features and statistics for a certain genome or all genomes that code a certain gene), links to the relevant FASTA and GenBank files are also made available.

BACA, although a very simple tool, proved to be a massive time saver. As an example, BACA can retrieve and organize the mitochondrial genomes for all *Vertebrata* (545 at the time of writing) in less than 1 h. Furthermore, as the operation, when performed manually, is very error prone, BACA assures a much higher data quality.

BACA can facilitate extending the research carried out to a certain subset of data to a much bigger subset (e.g. the research carried out on *Amphibia* in (Fonseca *et al.* 2006) can now be easily expanded to all desired subclasses of *Vertebrata*). It also facilitates broadening the scope of some comparative studies from species-wide genomic studies to class-wide ones.

The presentation aspects proved valuable both as an educational tool and also to get immediate insights regarding comparative genome lengths and gene recombinations.

Acknowledgements

A.B.-P. is supported by research grant SFRH/BPD/17822/2004, M.M.F. and D.J.H. are supported by POCI/TEN-BDE/61946/2004 both from Fundacao para a Ciencia e Tecnologia, Portugal, programme POCI 2010, cofinanced by FEDER.

4 PROGRAM NOTE

References

- Bazin E, Duret L, Penel S, Galtier N (2005) Polymorphix, a sequence polymorphism database. *Nucleic Acids Research*, **33** (Database issue), D481–484.
- Boore JL (1999) Animal mitochondrial genomes. *Nucleic Acids Research*, **27** (8), 1767–1780.
- Chenna R, Sugawara H, Koike T *et al.* (2003) Multiple sequence alignment with the CLUSTAL series of programs. *Nucleic Acids Research*, **31** (13), 3497–3500.
- Fonseca MM, Froufe E, Harris DJ (2006) Mitochondrial gene rearrangements and partial genome duplications detected by multi-gene asymmetric compositional bias analysis. *Journal of Molecular Evolution*, **63** (5), doi: 10.1007/s00239-005-0242-9.
- Hassanin A, Leger N, Deutsch J (2005) Evidence for multiple reversals of asymmetric mutational constraints during the evolution of the mitochondrial genome of metazoa, and consequences for phylogenetic inferences. *Systematic Biology*, **54** (2), 277–298.
- Korab-Laskowska M, Rioux P, Brossard N *et al.* (1998) The Organelle Genome Database Project (GOBASE). *Nucleic Acids Research*, **26** (1), 138–144.
- R Development Core Team (2005) *r*: A language and environment for statistical computing, Vienna, Austria. ISBN 3-900051-00-3.
- Rest JS, Ast JC, Austin CC *et al.* (2003) Molecular systematics of primary reptilian lineages and the tuatara mitochondrial genome. *Molecular Phylogenetics and Evolution*, **29** (2), 289–297.
- Stajich JE, Block D, Boulez K *et al.* (2002) The Bioperl Toolkit: Perl modules for the life sciences. *Genome Research*, **12** (10), 1611–1618.
- Xia X (2005) Mutation and selection on the anticodon of tRNA genes in vertebrate mitochondrial genomes. *Gene*, **345** (1), 13–20.

BACA: a mitochondrial genome retriever, organizer and visualizer

In this chapter relevant complementary information about BACA is provided. This is mainly due to the fact that application notes in most peer-reviewed journals have stringent limitations in size, thus making very difficult to include all the relevant information.

4.1 Architecture and implementation

BACA is a typical web client/server application, which can be accessed from a normal web browser (although SVG[22] support is highly recommended), its architecture can be seen on figure 4.1.

BACA has a typical, simple web frontend. After the user supplies a query to the frontend, that query is processed via an internal flow of very simple components. The components are divided in three classes: Interaction with GenBank, data organization and statistical calculation and generation of the interactive web interface.

The system was implemented in Perl, using BioPerl[20] to facilitate the interaction with GenBank and metadata (feature/annotation) processing.

In an attempt to accommodate all users we developed not only a highly interactive SVG interface, where the user can select which features to view and inspect details over a certain feature of a certain genome, but also two other simpler interfaces supporting non-SVG compliant browsers. These simpler interfaces have a cruder visualization tool but still allow the download of all organized data for use, locally, by the user.

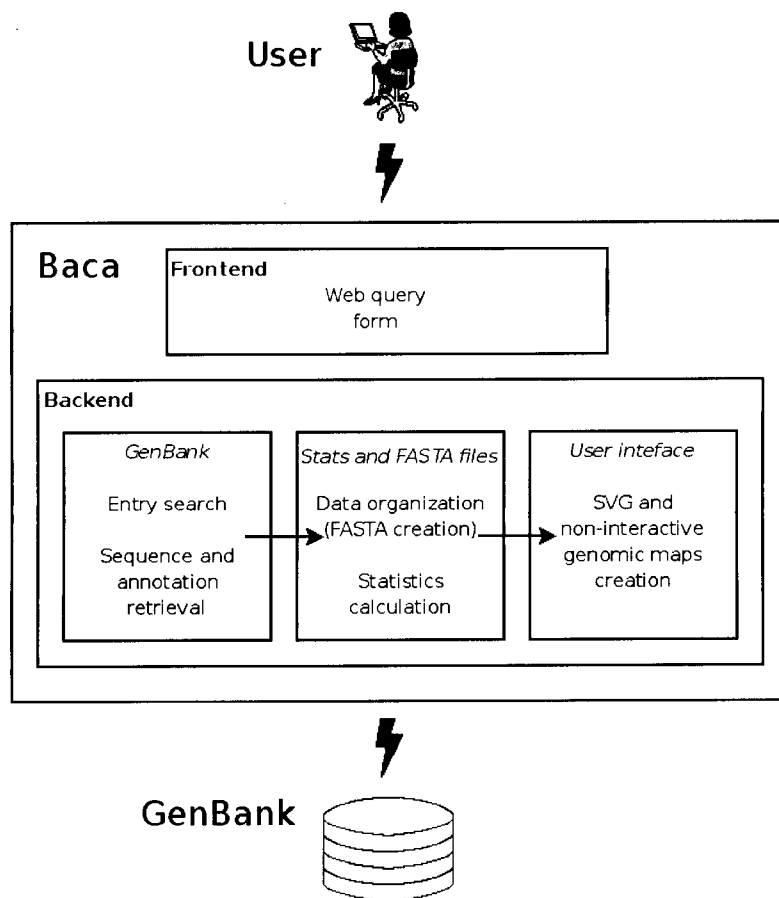


Figure 4.1: BACA software architecture

4.2 The GenBank interaction component

Here, the GenBank interaction component (the simplest of all components) is presented. The presentation serves two purposes: To discuss implementation details, and to demonstrate that, with the proper libraries, some tasks are quite trivial to code. The other two components are not presented as the biological relevant code goes along the same lines and there is quite some code that is not relevant in a biological setting (e.g., the SVG generating code).

The first module which does entry search in GenBank, retrieves all entry IDs which are the answer to the user supplied query. After that, the module to retrieve entries, gets each ID and retrieves the relevant entry.

Entry search in GenBank

```
#!/usr/bin/perl

$| = 1;
use Bio::DB::GenBank;
$gb = new Bio::DB::GenBank;                    5

my $query = Bio::DB::Query::GenBank->new
    (-query => $ARGV[1],
     -db    => $ARGV[0]);                      10

@ids = $query->ids;
foreach $id (@ids) {
    print $id, "\n";
}                                             15
```

As it can be seen, the code to search GenBank IDs amounts to almost nothing. Here follow some comments on the code lines, mainly targeted at readers that don't know how to code and would like understand and maybe reuse the code presented. Line 4 imports BioPerl's GenBank functions. Line 5 makes those functions accessible on the object `$gb`. Lines 7-9 perform the query on GenBank, the parameters for the query where supplied for the user and are stored on variables `ARGV[1]` (the user supplied query, e.g. *Amphibia and mitochondrion*) and `ARGV[0]` (the database to be searched, in our case NCBI's *genome* database). The remaining of the code just iterates through the answer (i.e., the ID list) from GenBank and prints the IDs (the printed values, will be used as input for the next module).

A note is due on the overall code and especially on line 3 (which sometimes is written `$|++`). This piece of code might be considered (and rightfully so, in my opinion) as cryptic, strange and unreadable. That is a feature of the underlying programming language Perl, and not a issue with programming per se. Further comments about programming languages and bioinformatics can be read on the discussion chapter.

Sequence and annotation retrieval from GenBank

```
#!/usr/bin/perl

use Bio::DB::GenBank;
use Bio::SeqIO;                                5

$gb = new Bio::DB::GenBank;

while ($id = <STDIN>) {
    $gb = new Bio::DB::GenBank;                10

    my $query = Bio::DB::Query::GenBank->new
```

```

(-ids => [$id],
 -db   => $ARGV[0]);

my $seqio = $gb->get_Stream_by_query($query);           15
while( my $seq = $seqio->next_seq ) {
  $sio = Bio::SeqIO->new(-file => $seq->id . '.gbk', '-format' => 'genbank');
  $sio->write_seq($seq);
  open $fh2, '>data/' . $seq->id . '.meta';
  print $fh2 $seq->desc, "\n";                         20
  close $fh2;
}
}

```

In this case line 8 iterates over all IDs supplied by the previous module. For each ID supplied the genome is retrieved (lines 11-13), its contents accessed (line 15), and for each sequence inside (line 16, only one sequence is inside, but the code supports queries for more than one sequence at a time), its text is retrieved in GenBank format (line 17) and written to a separate file (lines 18-21).

m4s2 - Article submitted

In this chapter the m4s2 article, submitted to *Bioinformatics* can be found.

Universidade do Porto
Faculdade de Ciências
DEPARTAMENTO DE
CIÊNCIA DE COMPUTADORES
BIBLIOTECA

modeler4simcoal2: A user-friendly, extensible modeler of demography and linked loci for coalescent simulations

T. Antao^{a,c*}; A. Beja-Pereira^a, G. Luikart^{a,b}

^aCIBIO, Centro de Investigação em Biodiversidade e Recursos Genéticos, Campus Agrário de Vairão, Universidade do Porto, Portugal

^bDivision of Biological Sciences, University of Montana, USA

^c Departamento de Zoologia e Antropologia, Faculdade de Ciências do Porto, Portugal

ABSTRACT

Summary: Modeler4simcoal2 (m4s2) is an extensible graphical tool to model linked loci and population demographies. M4s2 is easy to use, allowing for the modeling of complicated scenarios, making coalescent simulation modeling accessible to biologists with limited computer skills. The software includes an extension system allowing for new models to be created, published and downloaded from the Internet.

Availability: m4s2 is available from <http://popgen.eu/soft/m4s2> under a GPL license. The website also contains guides, screen shots and tutorials.

Contact: tra@mail.icav.up.pt

1 INTRODUCTION

Currently the use of the coalescent in population genetics is widespread (e.g., Voight *et al.* (2006); Akey *et al.* (2004)). Its popularity is largely due to its computational efficiency which allows for relatively rapid simulation of complex or large data sets (e.g., many loci and individuals). Simulations are mainly used to determine confidence intervals for statistics (e.g., F_{st} , Tajima D) and also to quantify the power and reliability of statistical methods (e.g., Tallmon *et al.* (2004)). For example a procedure to detect candidate adaptive loci is to simulate 1000s of neutral loci, compute summary statistics and, identify those genotyped loci that fall outside the simulated confidence intervals for neutral loci. The models used in most publications are simple, with at most two demographic events in a single model (e.g., bottleneck followed by a split in two populations). Although the models used are gross simplifications of reality, nothing is known about the behavior of relevant statistics if more complex models are created. More importantly, few studies have been made on the acceptability of model simplification, that is, if simple models can replace more complicated ones and still give similar confidence bands for important statistical indicators.

Most of the existing coalescent simulation programs (e.g., (Mallund *et al.*, 2005; Laval and Excoffier, 2004)) provide either a flexible but complex Domain Specific Language which require some programming skills that are beyond the abilities of much of the target audience or impose quite strict limitations on what can be simulated (Beaumont and Nichols, 1996; Hudson, 2002; Spencer and Coop, 2004) both for the demography and marker modeling. Even the applications imposing limitations are often not easy to use.

Also, importantly, it is easy to make errors on the modeling part (e.g., by wrongly calculating growth rates or migration matrices), errors that, contrary to a general programming language are hard to detect, thus generating erroneous results. Furthermore, in some simulators, creating scenarios with many populations is in practice impossible to do by hand as it implies the creation of very large matrices.

As the existing programs are generally difficult to use, only very simple scenarios tend to be modeled. This is probably one reason why few or no studies have evaluated the potential problems (e.g., biases) from using simplified models or why most research articles avoid complex models that represent reality more closely.

There are applications that facilitate the modeling of demographies and marker relationships: Coasim has a interface that allows modeling of marker relationships, but only one demography is possible, namely a population of constant size. Easypop (Balloux, 2001) is a forward-time simulator that has an easy to use command-line interface, it allows the simulation of several reference demographies. Easypop limitations are: a single recombination rate among physically adjacent loci, some more specialized demographies are either impossible or difficult to model, and, being a forward-time simulator, Easypop has all the advantages and disadvantages of this type of simulator namely forward-time simulators are computationally much more intensive than coalescent ones. In any case, Easypop easy to use command-line modeling facilities makes it the most direct comparison with our graphical modeler.

2 IMPLEMENTATION

In order to make a more user friendly system for modeling coalescent processes, we designed m4s2, an extensible graphical modeler for demography and linked loci. M4s2 is a Java Web Start application, allowing direct execution from the web. Currently the program generates input files to be run on Simcoal2 (Laval and Excoffier, 2004), a well established coalescent simulation program. The application is divided in two parts: one to model markers that are unlinked or physically linked in chromosome blocks, and another to model different population demographies.

The first issue that the application addresses is modeling chromosome blocks, that is, blocks of the genome that are independent from a linkage disequilibrium point of view. Each chromosome block has a set of markers that are in linkage disequilibrium. It supports both markers with only frequency information (SNPs, and RFLPs) and markers with genealogical information (microsatellites and DNA

*to whom correspondence should be addressed

sequences). The full expressive power of Simcoal2 is captured by m4s2's chromosome modeler (image available on the screen shot section of the web site), e.g. the Generalized Stepwise Mutation (GSM) model for microsatellites, which allows single or multistep mutations, is fully supported.

The second part of m4s2 deals with demography modeling. A set of predefined demographies are built-in (figure 1 shows nine different demographies), including, the many demographies that we found in published literature using coalescent simulations. A screen capture of modeling a bottleneck can be seen on the web site.

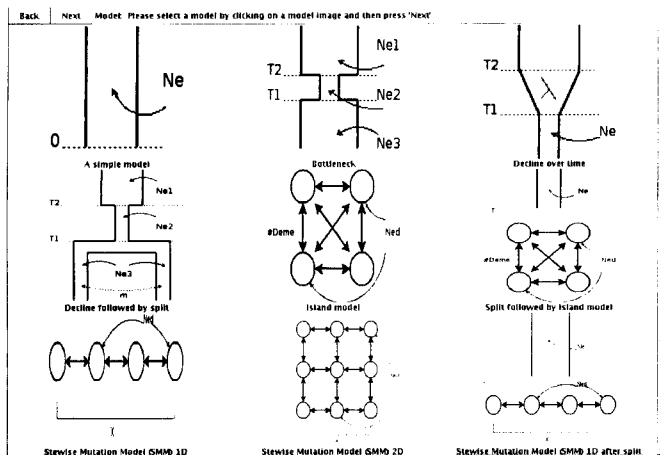


Fig. 1. Demography modeling in m4s2.

The demography modeler is fully extensible. An extension language was developed which can be embedded on Simcoal2 template files (which will be converted to the final Simcoal2 format before simulation). Also, a small parameter language is available allowing for specification of parameters and constraints (e.g., if two parameters T1 and T2 refer to time events then T1 has to be greater than T2). M4s2 will read the Simcoal2 template file, the parameter list and an supplied image illustrating the demography and automatically create a user-friendly interface for the new model. The system will not allow the user to violate the constraints imposed by the model designer.

If the template system is not enough to express the desired model then the template can still be extended in Jython (Python on Java), giving the model creator full expressive power to create new demographies. The extension system is targeted to the more knowledgeable user with regards to computer science. M4s2 is then capable of importing these foreign models directly from the web. We envision a scenario where a bioinformatician creates and publishes a set of new demographies and a population geneticist simply imports the new models directly from the web. This permits the separation of tasks between members of a team allowing for each one to concentrate on their core specialty. The application hides unnecessary complexity of the computational part, in as much as possible, for the population geneticist. In addition, we provide a set of extra models, which are were designed to study the spread of domesticated species from the Fertile Crescent across Europe and Asia.

After the modeling part is done, m4s2 will create Simcoal2 input files. M4s2 can then optionally be used to call Simcoal2 (to generate 1000s of simulated data sets) and after that Arlequin (Excoffier et al., 2005) for analysis of the simulation results (e.g., computation of F_{st}). M4s2 can be used stand-alone from Simcoal2 and Arlequin.

3 CONCLUSION

M4s2 will facilitate the study of more complex models, help increasing our knowledge about the reliability of simpler models, and increase the usage of coalescent simulation by lowering the barriers to usage of these type of applications.

In the future we intend to support more simulators (both coalescent and forward-time) giving the user choice on which application to use.

In an era where non user friendly software abounds, it is important to remove unnecessary complexity from the users. M4s2 does not intend to hide important modeling issues, on the contrary, by easing the burden on using a coalescent simulator, it will expose the user just to the fundamental modeling issues, thus emphasizing the importance of understanding the underlying theory. The tool also uses existing, well established, programs giving users not only easy to use interfaces for those programs, but also new functionality built on top of those applications, instead of trying to recreate functionality that is already available elsewhere. It is in this philosophy that m4s2 was built.

ACKNOWLEDGMENTS

AB-P was supported by research grant SFRH/BPD/17822/2004 and POCI/CVT/567558/2004 from Fundacao para a Ciencia e Tecnologia, Portugal. GL was supported by FLAD (Luso-American Foundation), UP, CIBIO, and UM.

REFERENCES

- Akey, J. M., Eberle, M. A., Rieder, M. J., Carlson, C. S., Shriver, M. D., Nickerson, D. A., and Kruglyak, L. (2004). Population history and natural selection shape patterns of genetic variation in 132 genes. *PLoS Biol.* **2**(10), e286.
- Balloux, F. (2001). EASYPOP (version 1.7): A computer program for population genetics simulations. *Journal of Heredity*, **92**(3), 301–302.
- Beaumont, M. and Nichols, R. (1996). Evaluating loci for use in the genetic analysis of population structure. *Proceedings: Biological Sciences*, **263**(1377), 1619–1626.
- Excoffier, L., Laval, G., and Schneider, S. (2005). Arlequin ver. 3.0: An integrated software package for population genetics data analysis. *Evolutionary Bioinformatics Online*, **1**, 47–50.
- Hudson, R. R. (2002). Generating samples under a wright-fisher neutral model of genetic variation. *Bioinformatics*, **18**(2), 337–338.
- Laval, G. and Excoffier, L. (2004). Simcoal 2.0: a program to simulate genomic diversity over large recombining regions in a subdivided population with a complex history. *Bioinformatics*, **20**(15), 2485–2487.
- Mailund, T., Schierup, M. H., Pedersen, C. N. S., Mechlenborg, P. J. M., Madsen, J. N., and Schausser, L. (2005). Coasim: a flexible environment for simulating genetic data under coalescent models. *BMC Bioinformatics*, **6**, 252.
- Spencer, C. C. A. and Coop, G. (2004). Selsim: a program to simulate population genetic data with natural selection and recombination. *Bioinformatics*, **20**(18), 3673–3675.
- Tallmon, D. A., Luikart, G., and Beaumont, M. A. (2004). Comparative evaluation of a new effective population size estimator based on approximate bayesian computation. *Genetics*, **167**(2), 977–988.
- Voight, B. F., Kudavalli, S., Wen, X., and Pritchard, J. K. (2006). A map of recent positive selection in the human genome. *PLoS Biol.* **4**(3), e72.

m4s2

In this chapter relevant complementary information about m4s2 is provided, again, like in the case of BACA, because the application note submitted is too short to document all the important details.

6.1 m4s2 components

As can be seen on the article, m4s2 has three parts: a demography modeler, a chromosome modeler and controller for Simcoal2 and Arlequin3.

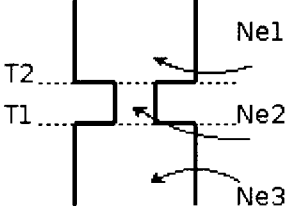
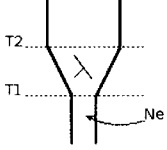
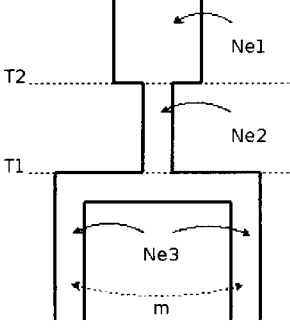
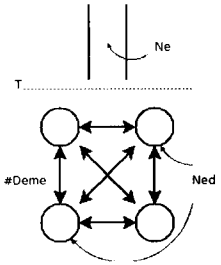
Both the chromosome modeler and the controller are useful, but have no fundamental feature that requires further analysis. Regarding the demography modeler below are presented some of the demographies that are built-in and the demography extension system.

6.2 Supported demographies

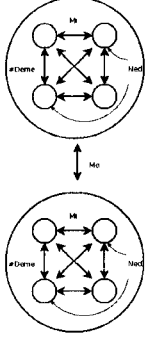
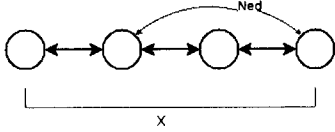
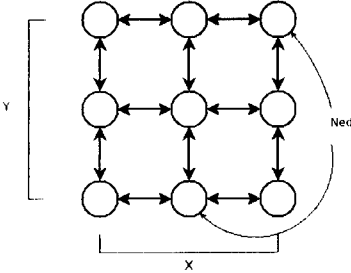
Here is presented a list of supported demographies, these are mostly the typical population genetics scenarios. When a less common scenario is presented, the rationale behind its creation is presented.

As the program is still evolving it is possible that the supported demographies change over time, most probably increasing in number.

ID	Figure	Description
1		The simplest model template possible is the one where there is no population structure and no demography events (population size is constant). In this case the only parameter is N_e .

ID	Figure	Description
2		<p>An immediate bottleneck (decline followed by expansion). The parameters in this case are: N_{e1} - Original population size, T_2 - Generation when the decline occurs, N_{e2} - Population size inside the bottleneck, T_1 - Generation when the expansion occurs, N_{e3} - Current population size. If $N_{e1} = N_{e2} = N_{e3}$ then we have scenario 1. If $N_{e2} = N_{e3}$ then we have only decline. If $N_{e1} = N_{e2}$ then we have only expansion.</p>
3		<p>A decline over time. The parameters in this case are: N_e - Current population size, T_2 - Generation when the decline starts, T_1 - Generation when the decline ends, λ - Decline rate.</p>
4		<p>An immediate decline followed by a split. The parameters in this case are: N_{e1} - Original population size, T_2 - Generation when the decline occurs, N_{e2} - Population size after the decline, T_1 - Generation when split occurs, N_{e3} - Size of each branch after the split, m - Symmetrical migration rate between demes. If $N_{e2} = N_{e3}$ then there is no decline.</p>
5		<p>The Island Model splitting from an original population¹. The parameters in this case are: N_e - Original population size, T - The generation when sub structuring occurs, #Deme - Number of demes, N_{ed} - Population size for each deme. m - Symmetrical migration rate among demes.</p>

¹The pure Island Model is also supported.

ID	Figure	Description
6		<p>Hierarchical population structure, where two metapopulations (i.e., each one is composed itself of a set of populations) exchange individuals. The parameters in this case are: N - Population size of each deme, $\#Demes$ - Number of demes inside each metapopulation, M_i - Migration inside metapopulations, M_e - Migration among metapopulations.</p>
7		<p>Stepwise Mutation Model (SMM), one dimension. The parameters in this case are: N_e - Original population size, N_{ed} - Population size for each deme, X - Number of demes (single dimension). m - Symmetrical migration rate among neighbouring demes.</p>
8		<p>Stepwise Mutation Model (SMM), two dimensions. The parameters in this case are: N_e - Original population size, N_{ed} - Population size for each deme, X - Number of demes ("horizontal" dimension), Y - Number of demes ("vertical" dimension). m - Symmetrical migration rate among neighbouring demes. The number of demes is $X * Y$.</p>

ID	Figure	Description
9		<p>An immediate decline followed by a split, where one of the sides is an island model and the other side a single population completely isolated (no gene flow). This scenario seems to be quite realistic and cause some selection detection applications to give erroneous results[27]. The parameters in this case are: N_{e1} - Original population size, T_2 - Generation when the decline occurs, N_{e2} - Population size after the decline, T_1 - Generation when split occurs, N_{isol} - Population size of the isolated population, #Deme - Number of demes in the Island, N_{ed} - Population size for each deme of the Island, If $N_{e1} = N_{e2}$ then there is no decline. m - Symmetrical migration rate in the island model.</p>

Table 6.1: Models and possible parameters

6.3 Creating new demographies

Here I explain how to extend m4s2 demography models. M4s2 supports a set of demographies, which, although covering most of the demographies found on publications, are far from covering all possible cases (which are, obviously, limited only by ones imagination). As such an extension system is provided, which allows for new models to be incorporated into m4s2.

The system was developed with the following in mind: Although m4s2 can be used by the computer illiterate user, the extension system is targeted to a more computer savvy type of user. The idea is that a more computer literate user creates new models, making them publicly available on the web for all users to use. M4s2 is able to directly import models from the web (we provide one example here). A user extending m4s2 needs to now how to code Simcoal2 models, to use a simple embedded language described below, and in extreme cases to code in Python. It is expected, in reality that most extension models are very easy to code, but the full language expressiveness of Python is available for the most complex cases.

This section is organized around examples. Three examples are presented, in increasing order of difficulty, introducing concepts, small domain specific

languages and Python extensibility. The last part of the document concerns with deploying models on the web.

A simple example

I start with the simplest example possible: a single, constant size population. The parameters for this model are simply the population size and the sample size.

Three artifacts have to be provided: An image, a model template and a parameter file. An optional artifact is a Python extension file which will only be used in the hard example.

The image

The image has to be called `image.png`, any size is acceptable as `m4s2` resizes it.

The properties file

Here is a parameter file which has to be called `properties`, followed by an explanation.

```
id=simple
name=A simple model
numParameters=2
1.name=Ne
1.desc=Effective population size
1.pythonName=pop_size
1.value=100
1.type=int
2.name=sample_size
2.desc=Sample size per population
2.pythonName=sample_size
2.value=60
2.type=int
```

`id` is an internal id, it should be unique (among all models) and composed of small letters, underscores and numbers.

`name` is simply the name that is presented to the user.

`numParameters` is the number of parameters that the user has to parametrize, in our case just the population size and the sample size.

We can then describe the parameters, each parameter has a slot (identified by a number), a name, a description (which will be shown to the user), a type (either float or int), a default value and a `pythonName`, which is simply the name that the parameter will have on the template file (see next subsection).

There should also be a constraint between sample size and population size (namely that the population size has to be bigger than the sample size), m4s2 allows this, but for the sake of simplicity, we will only present that in the next example.

The template/model file

The template file (model.par), is a SimCoal2 file with added macro features, some of the most simplest are shown on the first example (the reader can consult Simcoal2 documentation for information about the Simcoal2 format):

```
//Parameters for the coalescence simulation program : simcoal.exe
1 samples
//Population effective sizes (number of genes 2*diploids)
?pop_size
//Samples sizes (number of genes 2*diploids)
?sample_size
//Growth rates : negative growth implies population expansion
0
//Number of migration matrices : 0 implies no migration between demes
0
//historical event
0 historical events
```

This is a simple SimCoal2 file where the population and sample sizes are replaced by ?pop_size and ?sample_size. Whenever '?' followed by a pythonName (see parameter file explanation) is found it will be replaced by the value chosen by the user).

An intermediate example

In this example, we will introduce constraints and in-model code execution. We will not discuss image issues any further.

We will discuss a decline followed by a split as can be seen on figure 6.1.

The properties file

The fundamental part is at the end, where constraints are introduced.

```
id=decline_split
name=Decline followed by split
numParameters=7
1.name=Ne1
1.desc=Effective population size before decline
1.pythonName=ne1
```

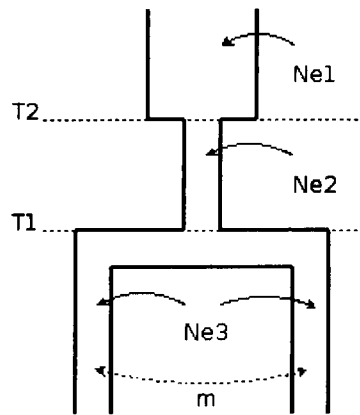


Figure 6.1: A decline followed by a population split

```

1.value=100
1.type=int
2.name=Ne2
2.desc=Effective population size after decline
2.pythonName=ne2
2.value=50
2.type=int
3.name=Ne3
3.desc=Effective population size of each split branch
3.pythonName=pop_size
3.value=100
3.type=int
4.name=T2
4.desc=Generation of decline
4.pythonName=contract_gen
4.value=50
4.type=int
5.name=T1
5.desc=Generation of split
5.pythonName=split_gen
5.value=25
5.type=int
6.name=mig
6.desc=Migration rate
6.pythonName=mig
6.value=0.01
6.type=float
7.name=sample_size

```

```

7.desc=Sample size per population
7.pythonName=sample_size
7.value=60
7.type=int
numConstraints=3
1.constraint=split_gen=ne2
2.message=Population size before decline has to be bigger or
equal to size after decline
3.constraint=ne2<=pop_size
3.message=Population size before expansion has to be smaller or
equal to size after expansion

```

`numConstraints` tells the number of constraints, in our case 3.

Then for each constraint, we have a condition (in the format of a Python conditional) and a message, which will be reported to the user in case she violates the condition, in that case the system is not allowed to proceed, until the constraint is satisfied.

The template/model file

This template file introduces embedded executable code, the fundamental part is at the bottom.

```

//Parameters for the coalescence simulation program : simcoal.exe
2 samples
//Population effective sizes (number of genes 2*diploids)
?pop_size
?pop_size
//Samples sizes (number of genes 2*diploids)
?sample_size
?sample_size
//Growth rates : negative growth implies population expansion
0
0
//Number of migration matrices : 0 implies no migration between demes
2
//mig
0 ?mig
?mig 0
//nothing
0 0
0 0
//historical event
2 historical events

```

```
?contract_gen 0 0 1 !!!1.0*?ne1/?ne2!!! 0 1
?split_gen 1 0 1 !!!1.0*?ne2/(2*?pop_size)!!! 0 1
```

All the text between !!! will be executed by the Jython interpreter, its result will replace the code on the template. Arbitrary functions can be called (see the next example).

The hard example

The next example is an (excessively) complicated one and can be seen on figure 6.2. It was developed as a model for the evolution of domesticated species. I recommend not spending too much time trying to understand the figure, the fundamental point is that it is full of demographic events and has hierarchical population structure and migration.

The properties file is not shown as it introduces no new concepts.

The template/model file

The template file is presented below, it introduces function calls, a simple concept. Some functions are not available on the core m4s2, so they will have to be provided (see next subsection).

```
//Parameters for the coalescence simulation program : simcoal.exe
!!!?pops_per_group*3!!! samples
//Population effective sizes (number of genes 2*diploids)
!!!generate_pop_sizes(?pops_per_group, ?pop_size)!!!
//Samples sizes (number of genes 2*diploids)
!!!dupe(str(?sample_size), ?pops_per_group*3)!!!
//Growth rates : negative growth implies population expansion
!!!dupe('0', ?pops_per_group*3)!!!
//Number of migration matrices : 0 implies no migration between demes
4
//multimig
!!!generate_goatsie_all_mat(?pops_per_group,?pop_size,?minternal,?me,?mimi)!!!
//mig3
!!!generate_goatsie_three_mat(?pops_per_group,?me,?mimi)!!!
//mig2
!!!generate_goatsie_two_mat(?pops_per_group,?me)!!!
//nothing
!!!generate_null_mat(?pops_per_group*3)!!!
//historical event
!!!?pops_per_group*3 - 3 + 5!!! historical events
4000 0 0 1 50 0 3
?expand_gen 0 0 1 0.2 0 3
?split_gen 1 0 1 2 0 3
```

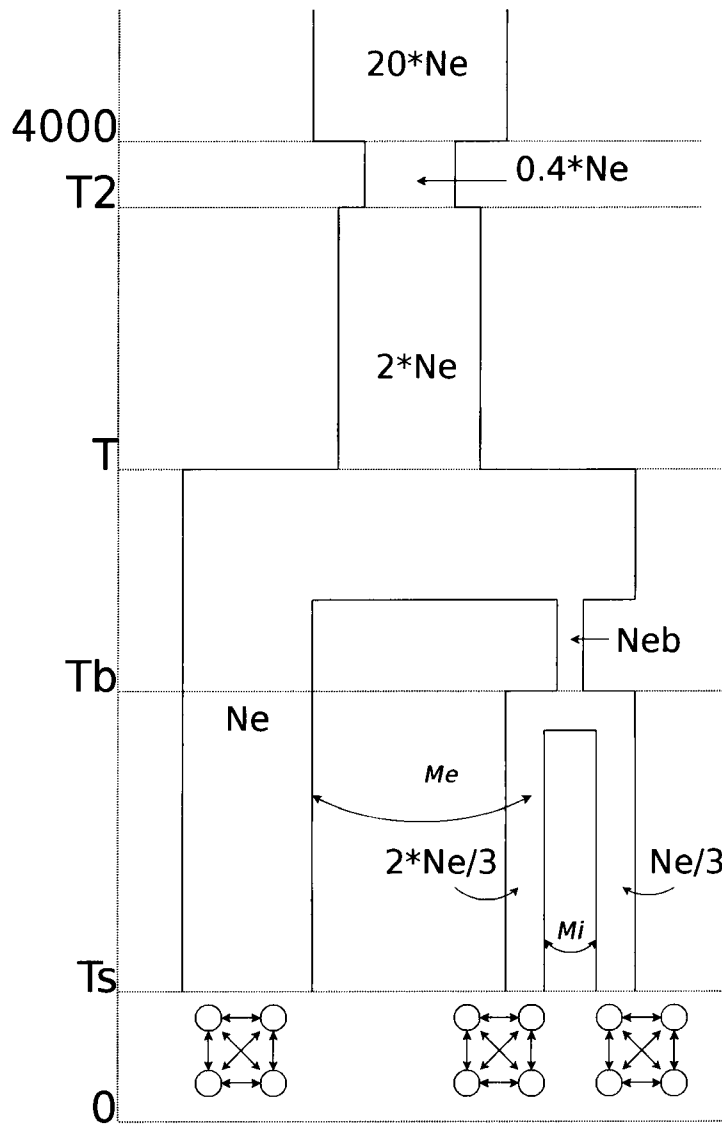



Figure 6.2: A complex demography

```
?split_europe_gen 1 1 1 !!!3.0*?neb/(2*?pop_size)!!! 0 2
!!!?split_europe_gen-1!!! 2 1 1 1 0 2
!!!generate_goatsie_sub_pop_events(?pops_per_group,?hier_split_gen)!!!
```

Some of the functions called are built-in on m4s2, the most useful case is `dupe`, which replicates a certain string a number of times (this is quite useful when the number of populations is a parameter, making the number of lines unknown on template construction).

Some others, like `generate_goatsie_all_mat` have to be supplied by the model maker.

The Jython extension file

In case of need an extension file (`extension.py`) can be created. This file can include any Jython code and do literally anything. It is expected that any code called returns a string (which will be replaced in the model).

Here is one of the functions used in this example:

```
def generate_goatsie_three_mat(pops_per_group, me, mi):
    goatsie_mat = ''
    total_size = pops_per_group*3
    for x in range(1, total_size + 1):
        for y in range(1, total_size + 1):
            if x==1 and y==2:
                goatsie_mat += str(me) + ' '
            elif x==2 and y==1:
                goatsie_mat += str(me) + ' '
            elif x==2 and y==3:
                goatsie_mat += str(mi) + ' '
            elif x==3 and y==2:
                goatsie_mat += str(mi) + ' '
            else:
                goatsie_mat += '0 '
        goatsie_mat += '\r\n'
    return goatsie_mat
```

This function generates a migration matrix among three metapopulations (each composed of an island model), the migration among metapopulations is not the same, being expected that the ones representing North European and Mediterranean areas have a bigger migration rate then the one between Mediterranean and East Asian areas.

Existing support functions

The following functions are available built-in (more can be added in the future if the seem to be useful in a wide scope):

`dupe(str,num_times)` duplicates a string a number of lines.

`generate_island_mat(total_size, mig)` generates an island matrix of `total_size` demes with migration `mig`.

`generate_ssm2d_mat(x_size, y_size, mig)` generates a Stepping Stone Model 2D matrix of `x_size`, `y_size` with migration `mig`. If `y_size = 1` then its an 1D matrix.

`generate_null_mat(total_size)` generates a matrix with no migration.

Packaging

After creating the models, these have to be packaged.

The first file to be created is the list of models, here is an example:

Model list file

Domestication and humans

http://popgen.eu/soft/m4s2/ext_models/goat1/

http://popgen.eu/soft/m4s2/ext_models/goat4/

The first line has to be Model list file

The second line is an human readable identifier

Each extra line points to an URL where a model can be found, on that URL the files properties, model.par, image.png, extension.py (the last one being optional) have to exist.

This list file has to be put somewhere on webserver. The URL to the list file is the one to be supplied to m4s2.

Discussion

7.1 Comments on the software developed

Bairom

Bairom was started before I took a course in Population Genetics. The only two courses then taken were Introduction to Genetics and Genomics, and Molecular Markers. As such, and as an example, the data representation of synthetic genomes as been changed countless times. From a first version where I thought that microsatellites could be inside exons to the current one, a lot as changed. In fact the knowledge representation of what is important in a population genetics setting is a hard problem. Mixing different types of markers, how to represent positional information, how to handle linkage disequilibrium are not easy to answer questions if one is trying to make a *all size fits all* simulator.

Other non trivial questions would be, among others, metapopulation management, geographic information (which could be useful in a landscape genetics setting [16]) and genotypical/phenotypical relationships. Engineering questions are also not to be forgotten, for instance a implementation with sophisticated recombination would be much slower than one with support for only independent loci. Independent loci simulation can still be used in most animal genetics' cases thus, such a simulator, while more restricted would still be useful and much more performant.

There is a conscious decision of not to continue on this path. Extending and maintaining Bairom is mainly a programming exercise, which is not of really the main focus of interest of the author. A much preferred alternative is to help maintain an existing simulator, like Nemo [11], by supplying functionality that might be still missing. There are also philosophical reasons for such a decision: There is clear preference for not "reinventing the wheel" and to invest more in a culture of contributing and sharing.

BACA

BACA grew from tackling a clear problem: most of the time spent on analyzing mitochondrial genomes was in downloading and organizing sequences from GenBank. This was a clear waste of human resources, which can easily be solved by automating the process of downloading and organizing (this is mainly splitting each whole genome sequence in composing genes and supplying a file per gene with the said gene retrieved from all genomes). As such a small application was developed to do perform this task. The original program had less than 50 lines of code and was able to automate the majority of the manual tasks. After that, a second part was developed to calculate simple statistics on the sequences downloaded (e.g., strand compositional bias, in the form of GC skew), again the full application was less than 100 lines of code.

There is, in my opinion, a lesson to be learned for this exercise: if biologists had some minor knowledge of programming there would likely be many tasks that the gains from programming and automation would surpass by orders of magnitude the time spent in learning the basic programming skills needed to do that automation. In this case, the time spent on data downloading and organizing can be measured in a few weeks, against less than one day in developing a simple script to do all the work.

As its widely known by software engineers, going from a simple script to a fully polished (i.e., publishable) application takes quite some time: error detection in code can take up to 50% of the development time, furthermore there is documentation to be produced, testing to be done, extra functionality to make the application more *sexy*. For instance, in the case of BACA the visualization part was mainly added to insure the publishability of the application, this represented more than half of the work. Furthermore supporting all widely used browsers and platforms (Internet Explorer, Firefox, Safari on Windows, Mac OS X and Linux) required in practice having slightly different versions, all off which had to be developed, tested and now, maintained.

m4s2

As referred above, error detection code can take quite a big share of software development time. When we look at applications like Simcoal2 or fdist we easily notice that they are not very robust to user errors. Many bioinformatics applications, when presented with a user error simply crash, take all CPU time and normally don't supply useful output as to the causes of mistakes.

This is one of the reasons that makes m4s2 useful: it protects the user from the lack of resiliency of the wrapped simulator. Another advantage is that the complexity and difficulty of creating models for Simcoal2 is vastly reduced, this allows for non computer savvy users to use coalescent simulators or for them to have the *courage* to use more complex models. There is also another, qualitative advantage of this application: models with many popula-

tions become possible as it is in practice impossible to manually create input files for Simcoal2 with 100 populations as that would require at least a matrix with 10.000 cells.

There are three important lessons to be taken from m4s2:

First, it is a program that uses other programs. It builds on others' work and improves it. This is a clearly different approach from what is common practice in bioinformatics: isolated applications, that might be connectible with manual intervention, but which normally don't work simultaneously together either to facilitate the user experience or to provide conceptually new functionality. One notable exception happens with Serial Simcoal[1], which is an extended version of Simcoal supporting sampling in multiple points in time. It is just coincidence that Simcoal was used in both cases.

Second, m4s2 is really a byproduct, it just happened because there was some other work (namely generating coalescent models for robustness testing of selection detection methods) which had code that could be used for this purpose, namely a software library that could do parametrized model generation.

Third, clearly, doing polished web user interfaces has a low "quantity of work" to "results observed" ratio. Not only the quantity of code is immense (m4s2 is much bigger than, e.g., fdist2) as the direct scientific value of the application is quite low, i.e., their purpose is to help users using (hard to use) applications, and not doing analysis.

7.2 Sociology of bioinformatics

Atoms live alone, molecules are teams of atoms. In real life atoms alone are not that important, even when we talk about, say, Oxygen, we are normally talking of O_2 , a molecule. It was quite frustrating when people behave as if they were alone in the world.

The way of thinking, the problems, the day to day life of the researchers involved in bioinformatics is quite different, varying a lot between scientific areas. There are many ways a multidisciplinary team of researchers can fail to communicate. When there is no experience at all in multidisciplinary communication the first attempts will not be very productive. Furthermore, a stance, which sometimes happens, where one looks at his/her own background as more important could be a recipe for disaster.

The fundamental issue, I think, is not really understanding how others think (as that is quite hard, being really the *end of the line*), but a predisposition to accept the idea that others are different, that some effort has to be done just to communicate, that the cultural differences are so many that it will be difficult. Furthermore, some modesty when dealing with different scientific backgrounds is helpful. This would be the bare minimum for a multidisciplinary team to work effectively. Some curiosity for the work and mentality

of different approaches would also help. Curiosity coupled with the idea that it will take time and effort just to understand how others think and function. With time (years) and patience, real mutual understanding, where people can predict how others think and operate can develop. Luckily, people will sometime look and see that they gained from this multidisciplinary exchange of ideas, then just *staying home*, just interacting with *their own kind*.

7.3 Making software for bioinformatics

A few issues deserve being mentioned about software creation in a bioinformatics setting, namely: The role of software creation in a research programme, what type of software should be constructed especially in the light of existing applications and technical considerations on how software should be done.

The role of software creation in a research programme

What should be the importance of software creation in a long-term individual research path? That answer will obviously vary from case to case but it would be important to note that, in bioinformatics, computer science and statistics are means to an end, not the end in itself. Although there is space for many research paths centered on software and statistics that would probably not be the best solution for someone whose origin is computer science, as that would entail little exposure the Life sciences part of bioinformatics. As such software development should be a subsidiary activity, supporting a main research focus in a Life sciences problem.

This would suggest a different approach to the publishing cycle from the one taken in this work. Publishing an application note should be the *end of the road*, not the beginning. As such, firstly some new research concept would be created and applied to a real problem, this would mean that the first public, published product of a research activity would be the analysis (or solving) of a real biological problem, the second step would be polishing and publishing the theoretical methodology underlying the analysis and the final step would be making available to the community the application implementing the new concepts and analysis proposed.

This approach relates well with the typical software development process where firstly a rough prototype is developed (and used in internal analysis) and only after that a production quality version is edged out and released to the public (with the corresponding applications note).

Deciding which program to create and the current software ecology

Looking at the current software ecology in bioinformatics can be quite a disturbing experience. A vast quantity of software exists, with quite a lot of

overlap among applications. A completely different strategy can be taken: to use existing tools and either improve them (an example of this strategy might be Serial Simcoal) or build conceptually new functionality on top of them (an example is m4s2, where a GUI/modeler is built on top of a coalescent simulator. Another example would be using existing coalescent simulators inside a new tool that would generate and test synthetic datasets for selection).

The suggestion is, thus, a component architecture where new functionality, new concepts, would be implemented on top of existing ones. This is in stark opposition with the current *reinvent the wheel* philosophy that seems to be highly pervasive in the field. A few stumbling blocks to advance this approach can be found:

1. Some existing software is not amenable to being componentized (e.g. Arlequin3). Fortunately there is enough software that can be reused either by being available as a library (e.g. CoaSim[15]), or, more commonly because it can be easily called for other programs (e.g., fdist2, Phylip[8]).
2. The license for some software might not be compatible with the whole idea of componentization (e.g. PAUP*[23])
3. The culture of the software authors might not be as open as the software licenses used in the software. This was a problem directly felt by the author.

In this context there is nonetheless space to start constructing more high level applications, that is applications that tend to solve more sophisticated problems, e.g., programs that do robustness analysis on methods (e.g., by using already existing simulators and software implementing methods) or decision support systems (by making inferences on the results of lower level tools).

Another fundamental question is what type of user is targeted? Many life sciences users have very little knowledge of computers both from an programming and also user perspective. Furthermore some exhibit horror at the idea that they have to think about computational issues. There is clearly a need for life scientists to learn a bit of computer science. It is interesting to compare computer science with mathematics, whereas mathematics starts being learnt in kindergarten, little or no computer science training is provided (not to be confused with training as a user of computer applications like spreadsheets or word processors). Suggesting that most future life scientists will need to know about some programming as they do know (or should know) mathematics is probably a reasonable idea. In any case, in the current state of affairs there is clearly a *market* for easy to use applications or applications that facilitate the user of other applications, this is in fact the case of m4s2 which is mostly (but not only) a easy to user interface for Simcoal2. Two final points about this issue would be:

1. A life scientist with absolutely no knowledge of mathematics/computer science is gearing towards extinction.
2. It would be better if software authors wouldn't be proud of making software that is *difficult* to use. Especially because some of them reveal to be quite amateur at the art and science of software engineering, requiring from their users skills that they hardly possess.

To sum it up, depending on the type of application clear decisions will have to be made regarding making easy to use applications (which take time and are tedious to construct) or rather invest in more powerful frameworks which can accomplish more but will have a smaller audience. Careful analysis will have to be made on a case by case basis.

Technical issues

Quality assurance

Software engineering is a widely researched and widely used scientific and industrial discipline (software controls the possible mortgage on your home, your bank account, the computer where this text was typeset, most probably part of your car, the appliances of the GPS system used by field scientists to geo-reference their data, the system that will help doing a life saving surgery to you, the Internet, ...). It is characterized for being much more difficult to construct than most outsiders think it is and in fact its quality is very hard to access in most cases. Formal software testing is a costly operation normally only done in very critical systems. Another approach, software unit and integration testing is a branch of software engineering that is widely unknown to biologists that decide to make software although it is widely used in the industry. Of course testing, consumes time and seems not to be needed to assure publication. Of course, the means used should be proportional to the results expected, and, while most biological applications will not have the highly visible consequences of poor software engineering practices like the ones that led to the accident of Ariane 5[13], they are still used in such relevant fields as drug design or conservation biology.

Tools like coalescent simulators (among many others), are exposed not only to typical software bugs (like memory leaks) but also to very difficult to detect "biology" bugs, one example might be making a mistake on modeling a bottleneck event while coding a coalescent simulation and instead make an expansion. Such bugs tend not to make the program to stop and can easily go undetected as they produce slightly different results that might be acceptable under a *naked eye* examination.

The problem will only be solved when peer reviewed journals only accept applications where a standard test procedure was applied, until then, the quality of most available applications has a big question mark on top of

it. Talking informally with the authors of some existing applications only reinforces the idea that ignorance about testing is paramount and disdain for quality is almost explicitly as it is not externally enforced.

Openness

Although I would be personally inclined to do so on a moral and ethical base, I will not argue for free software on those grounds. Instead that will be argued mainly on pragmatic advantages for the scientific process. It is fundamental for researchers to be quite sure that the software that they are using works well, this means a combination of the following strategies should be used:

1. As stressed before, peer reviewed journals should enforce some kind of standard (and publicly available) test procedure for software.
2. If the source is available for inspection then users with the knowledge could directly inspect the code.
3. Programs should be possible to instrument in a way that they could be tested, i.e., by allowing for some form of automation on accepting some input and validating the output with an external tool.

This approach does not really force a free software license, but the code should still be available for public inspection. Of course, making the code available would make a *private* competitive advantage to fade away, although one must ask for the morality and legality of personal or institutional appropriation of a piece of work which is normally funded with public, tax-payers, money. The same line of reasoning is valid, by the way, for *my* datasets (*my* being a term usually coined by field biologists).

Programming languages

The work developed for this thesis was made in the programming languages Caml, Java, Jython (Python on Java) and Perl. R was also used. Furthermore BioPython[5] and BioPerl were also used. One can argue that using many tools is a proof of maturity and adaptation and, although there is some validity to that assertion, being highly proficient with a smaller set of languages and libraries might be a productivity booster, allow for the creation of a personal large library of utility functions and hold strong community bindings with the life sciences communities using the same tools.

For future work there was the decision to mainly concentrate around Python (with some possible work done in Java, R, C or Fortran if absolutely necessary). Below is a list of languages and programming environments along with a short, informal, highly personal assessment. The two main vectors of analysis are language elegance and suitability of libraries (not only in bioinformatics but also on producing graphics and generic scientific computation).

Perl Has probably the best library for biology: BioPerl (maybe with the exception of Bioconductor[10]/R). But Perl, as a language is unacceptable. Highly referred as a *write only* language, it is feasible to use for very small projects. It has a clear potential of damaging the reasoning of beginner programmers. Its convenient hacks are a chaos in the long run. The idea of a programming language where each one chooses its personal style is very poorly implemented. In fact, the Perl 6 effort seems to be an effort to render that philosophy in a correct way, in a more Domain Specific Language[26] approach. The expected approach to Haskell syntax and semantics on version 6 will be quite welcome. When version 6 is out, some reassessment might be in order. The fact that it does not run on a convenience virtual machine (JVM or CLR) is also a drawback.

Caml A modern functional language[12] done in a pragmatic way. With a reasonable sized community. Known not to compromise performance in spite of being functional and declarative. Unfortunately the libraries, both graphical and biological still lack.

Prolog Calling the language dead for practical purposes will not be far from the truth. Nonetheless logic programming and constraint logic programming would have much to offer in building decision support and automated analysis applications. Embedded engines (for virtual machines) like tuProlog[6] might be of use in future, more sophisticated applications.

R and bioconductor A statistical programming environment with extensive support for biology. The large offering in biology libraries and statistical tools is surely to be noted, and might be required in some applications. Nonetheless the underlying functional framework, while elegant lacks performance and the ability to construct easy to use user interfaces is restricted, in fact R not being a general programming language is also seen as a risk factor in the long term.

Java Widely available, has the support of modern and mature software engineering tools, make platform independent development really possible, and, contrary to popular belief has very fast implementations (when compared to, e.g., R or Python). Furthermore with technologies like applets or Java Web Start, Java applications are very easy to deploy for users with little computer literacy. Reasonable libraries exist. The fundamental issue with Java is the verbosity of the language and the fact that the pervasive culture is to make everything with a steep learning curve based on the idea that everything should be sacrificed to *pure* object-oriented design, this can even be seen in libraries like BioJava[19], where simple tasks like retrieving sequences from GenBank can take dozens of lines of code. Java clearly is not suited to be a prototyping

language, so a prototype to final product development approach cannot be taken. The fact that the JVM can accommodate other languages like Python or Ruby makes the platform still attractive. M4s2 was made as a Jython/Python hybrid and runs under the JVM.

Python Python is an agile, modern general-purpose language. Scientific libraries, with BioPython included are quite reasonable. Python can both be embedded in JVM and CLR virtual machines and has a bridge to R. The fundamental issues are lack of support for the creation of Domain Specific Languages (which can be found Python's current main competitor, Ruby) and very low performance.

C/Fortran Both languages lack of declarativity and closeness to the hardware paradigm make productivity and elegant programming quite difficult. However, abundance of scientific libraries and the ability to take the most performance of the hardware make them a possibility to implement the inner loops of computationally intensive programs (whereas all other code would be implemented in a higher level language).

Commercial offerings With the current quality offerings in the free and open source world, there was no need to research closed/for pay options.

7.4 Final comments

This discussion has been more on the informatics part than the biology part of bioinformatics, hopefully that means that all the issues pertaining to software are reasonably resolved and that the work that follows, that long research path which was decided after experimentation in this MSc can concentrate on what the author is really interested: going deep in the life sciences part of bioinformatics. Informatics and mathematics as a subsidiary, yet very important, tool to study life.

Lets then, put this Masters in bioinformatics to rest and let the really interesting and fun part begin...

Bibliography

- [1] C.N.K Anderson, U. Ramakrishnan, Y.L. Chan, and E.A. Hadly, *Serial SimCoal: a population genetics model for data from multiple populations and points in time.*, *Bioinformatics* **21** (2005), no. 8, 1733–1734.
- [2] T. Antao, A. Beja-Pereira, M.M. Fonseca, and D.J. Harris, *BACA: a mitochondrial genome retriever, organizer and visualizer*, *Molecular Ecology Notes* **7** (2007), no. 2, 217–219.
- [3] F. Balloux, *EASYPOP (version 1.7): A computer program for population genetics simulations*, *Journal of Heredity* **92** (2001), no. 3, 301–302.
- [4] M.A. Beaumont and R.A. Nichols, *Evaluating loci for use in the genetic analysis of population structure*, *Proceedings of the Royal Society B* **363** (1996), 1619–1626.
- [5] B. Chapman and J. Chang, *Biopython: Python tools for computational biology*, *SIGBIO Newsl.* **20** (2000), no. 2, 15–19.
- [6] E. Denti, A. Omicini, and A. Ricci, *Multi-paradigm Java-Prolog integration in tuProlog*, *Science of Computer Programming* **57** (2005), no. 2, 217–250.
- [7] A. Estoup, P. Jarne, and J.M. Cornuet, *Homoplasy and mutation model at microsatellite loci and their consequences for population genetics analysis*, *Molecular Ecology* **11** (2002), no. 9, 1591–1604.
- [8] J. Felsenstein, *Phylip - phylogeny inference package (version 3.2)*, *Cladistics* **5** (1989), 164–166.
- [9] M.M. Fonseca, E. Froufe, and D.J. Harris, *Mitochondrial gene rearrangements and partial genome duplications detected by multi-gene asymmetric compositional bias analysis*, *Journal of Molecular Evolution* **63** (2006), no. in press, 654–661.
- [10] R.C. Gentleman et al., *Bioconductor: open software development for computational biology and bioinformatics.*, *Genome Biol* **5** (2004), no. 10, R80.

- [11] F. Guillaume and J. Rougemont, *Nemo: an evolutionary and population genetics programming framework.*, *Bioinformatics* **22** (2006), no. 20, 2556–2557.
- [12] INRIA, <http://caml.inria.fr>, The Caml language.
- [13] P.B. Ladkin, *The Ariane 5 accident: A programming problem?*, Tech. report, University of Bielefeld, 1998.
- [14] G. Laval and L. Excoffier, *SIMCOAL 2.0: a program to simulate genomic diversity over large recombining regions in a subdivided population with a complex history.*, *Bioinformatics* **20** (2004), no. 15, 2485–2487.
- [15] T. Mailund, M.H. Schierup, C.N.S. Pedersen, P.J.M. Mecklenborg, J.N. Madsen, and L. Schauer, *CoaSim: a flexible environment for simulating genetic data under coalescent models.*, *BMC Bioinformatics* **6** (2005), 252.
- [16] S. Manel, M. Schwartz, G. Luikart, and P. Taberlet, *Landscape genetics: combining landscape ecology and population genetics*, *TRENDS in Ecology and Evolution* **18** (2003), 189–197.
- [17] R. Nielsen, *Molecular signatures of natural selection.*, *Annual Reviews in Genetics* **39** (2005), 197–218.
- [18] B. Peng and M. Kimmel, *simuPOP: a forward-time population genetics simulation environment*, *Bioinformatics* **21** (2005), no. 18, 3686–3687.
- [19] M. Pocock, T. Down, and T. Hubbard, *BioJava: open source components for bioinformatics*, *SIGBIO Newsletter* **20** (2000), no. 2, 10–12.
- [20] J.E. Stajich et al., *The bioperl toolkit: Perl modules for the life sciences*, *Genome Research* **12** (2002), no. 10, 1611–1618.
- [21] A. Strand, *Metasim 1.0: an individual-based environment for simulating population genetics of complex population dynamics*, *Molecular Ecology Notes* **2** (2002), 373–376.
- [22] W3C SVG Working Group, <http://www.w3.org/graphics/svg/>, Scalable Vector Graphics (SVG).
- [23] D. L. Swofford, *PAUP*: Phylogenetic analysis using parsimony (* and other methods)*, Sinauer, 2002.
- [24] D.A. Tallmon, G. Luikart, and M.A. Beaumont, *Comparative evaluation of a new effective population size estimator based on approximate bayesian computation.*, *Genetics* **167** (2004), no. 2, 977–988.
- [25] R Development Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2005, ISBN 3-900051-07-0.

- [26] A. van Deursen, P. Klint, and J. Visser, *Domain-specific languages: An annotated bibliography*, SIGPLAN Notices **35** (2000), no. 6, 26-36.
- [27] R. Vitalis, K. Dawson, and P. Boursot, *Interpretation of variation across marker loci as evidence of selection.*, Genetics **158** (2001), no. 4, 1811-1823.

