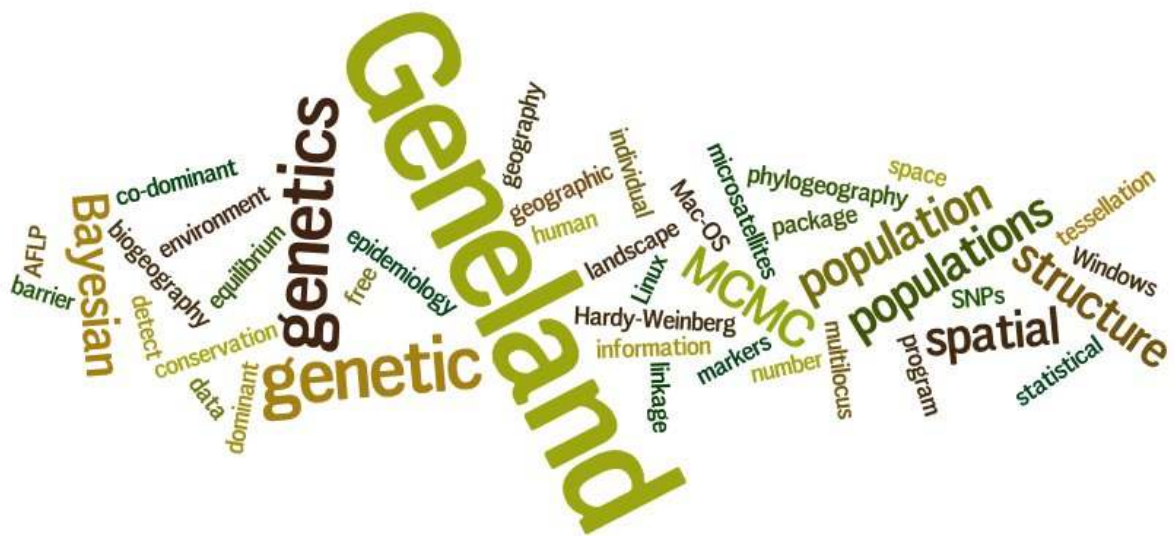# Population genetics analysis
using R and the GENELAND program

Gilles Guillot, Filipe Santos, Arnaud Estoup

February 15, 2012

Contents

1 Overview

1.1 About GENELAND

1.1.1 How to request help

If you are experiencing troubles with GENELAND, before requesting help, please

- make sure you are using the latest versions of R and GENELAND (version numbers appear at loading)

- read carefully the present documentation

- make sure you are able to describe and reproduce the problem you are experiencing

If you think you have identified a bug or feel the manual lacks information, you are welcome to report it, then

- make sure you are a registered GENELAND user

- send enough data and information about what you did in R for us to reproduce the problem, namely

  - on which operating system(s) did you experience trouble
  - if your problem is related to the use of GENELAND with the R command line, give the exact sequence of commands that causes the problem
  - if your problem appears with the GUI, please generate and send the `ExecLog` file located in the output directory (activate `Create Log file` option in top-left `Menu` panel.
  - in any case, send a subset of your data that will allow us to reproduce the problem

Questions should be addressed to Gilles Guillot: `gigu[à]imm.dtu.dk`.

1.1.2 History

The work around GENELAND started in 2002-2003 at INRA in Avignon, France. This resulted in an algorithm to perform clustering under a spatial as well as a non-spatial model for codominant markers [Guillot et al., 2005a]. This algorihtm has been made available as a regular R package officially released in 2005 [Guillot et al., 2005b]. Filipe Santos joined the project in 2007 and wrote the graphical user interface in R-Tcl/Tk together with Arnaud Estoup. People who helped designing and improving the package includes: Annie Bouvier, Aurélie Coulon, Jean-Francois Cosson, Frédéric Mortier and Benjamin Guedj.

Subsequent developments include a scheme accounting for the presence of null alleles [Guillot et al., 2008], an improvement of the inference technique under the correlated frequency model [Guillot, 2008], a more efficient MCMC post-processing scheme [Guillot, 2008], an option to handle dominant markers [Guillot and Santos, 2010] and a model to estimate the location and shape of hybrid zones [Guedj and Guillot, 2011].

The program and its manual benefitted from comments from users registered on the GENELAND mailing list and also from discussions with people who attended courses in populations genetics based on GENELAND in Lisbon, Prague and Porto in 2008 and in Helsinki and Oulu in 2009. They are all warmly thanked.

The GUI in R-Tcl/Tk is currently maintained by Filipe Santos and the R and Fortran codes are maintained by Gilles Guillot.

1.1.3 Credit

The GENELAND project received support from Institut National de le Recherche Agronomique, Bureau des Ressources Génétiques, Agence Nationale de la Recherche in France, the Norwegian Research Council, the Centre for Ecological and Evolutionnary Synthesis at the University of Oslo in Norway, the Technical University of Denmark in Copenhagen and the Danish Centre for Scientific Computing.

1.1.4   Contact, info, mailing list

GENELAND is distributed through the Comprehensive R Archive Network (CRAN). It consists of a network of mirroring sites throughout the world. This distribution method is very efficient but does not allow one to know how many users have downloaded or used a specific package. In order for people developing GENELAND to have an idea about that and also for users to be informed of updates and related publications, a mailing list is operated. Please register on `http://www2.imm.dtu.dk/~gigu/Geneland/register.php`.

Reports of bugs in GENELAND and errors in the present manual are greatly appreciated.

1.1.5   Citation

Developing, improving and maintaining GENELAND represents a tremendous amount of work. If you use it for your own scientific work, please cite the related publications (Guillot et al. [2005a],Guillot et al. [2005b],Guillot et al. [2008],Guillot [2008], Guillot and Santos [2010], Guedj and Guillot [2011]) detailed in the reference list at the end of the present document. For other references, type `citation(''Geneland'')` in the R prompt or see the GENELAND homepage `http://www2.imm.dtu.dk/~gigu/Geneland/`

1.2   System and hardware requirements

1.2.1   Operating system

GENELAND is an add-on to the free statistical software R. To install and run GENELAND, you need first to have R installed on your computer. R is available for MS-Windows, Linux and Mac-OS.
See `http://cran.r-project.org`. It is becoming a standard in many research communities, in particular in bioinformatics.
See `http://en.wikipedia.org/wiki/R\_(programming\_language)` for details.

1.2.2   Memory

Computations in GENELAND are carried out through a so-called Markov Chain Monte-Carlo (MCMC) technique. This implies that the overall computing task consists of a (very) long sequence of rather simple tasks. A small set of variables is stored in RAM, updated sequentially and written to the disk from time to time. This requires a few Mb of RAM. The exact amount varies with datasets and computing options but it is fulfilled by any current computer.

1.2.3   Disk space

The amount of disk space required depends on which fraction of the computations are stored on the disk. This amount can be fairly large from a few tens of Mega bytes to several Giga bytes (see section A.13 for details).

1.2.4   Computer speed

The model implemented in GENELAND is fairly complex. A run of 100000 iterations for a dataset of 200 individuals at 10 loci) takes typically 3-15 minutes with a computer equipped with a 2 GHz chipset. This time depends on the model and storage options.

1.3   Installation

1.3.1   Installing R

Instructions for installation of R are continuously updated.
See `http://cran.r-project.org/sources.html` for details and update.
Typically, under Windows:

- Go to the Windows binary repository `http://cran.r-project.org/bin/windows/base/`

- Download the executable R-x.x.x-win32.exe

- Launch this executable.

Useful sources of information include the various R manuals [1] and [Paradis, 2005, 2006] among others.

1.3.2   Installing Geneland

Once R is installed,

- launch R

- type `install.packages("Geneland")`.

- follow the instructions.

Under Mac-OS, make sure that X11 is installed and running when you install and use R. X11 can be found on the Mac-OS installation DVD.
For Tcl/Tk libraries (needed if you use Geneland) see the tools directory [2] and install Tcl/Tk 8.5.5 for X11 (tcltk-8.5.5-x11.dmg).

1.4   Tasks performed

1.4.1   Estimating the number of panmictic groups and locating their spatial boundaries

The main task of Geneland consists in clustering a sample into a certain number of groups in such a way that each group is homogeneous. Since its earlier version, Geneland can process population genetics data, i.e. genotypes at several loci. For usch data, the main assumption is that the putative groups are approximately at Hardy-Weinberg equilibrium with linkage equilibrium between loci (HWLE).

Since version 4.0.0, Geneland includes a model to deal with phenotypic data. See section 2.2 and Guillot et al. [2012] for detail.

Different algorithms based on different models are implemented. The most popular algorithm is based on a spatial model and makes use not only of genotypes but also of spatial coordinates of sampled individuals (or populations).

1.4.2   Input files

The research project that lead to the development of Geneland was focused on the combined use of genetics and geographic information to understand the factor affecting gene flow across space. Hence, a typical dataset treated by Geneland consists of

- a file containing the genotypes of n haploid or diploid individuals at L co-dominant markers (micro-satellites, SNPs);

- a file containing the spatial coordinates representative of each individual.

This second file is actually optional and Geneland can also be used without spatial information. See section 3 for detail about data format. As of version 4.0.0, Geneland can also process phenotypic data. The phenotypic data file should have one line per individual and one column per phenotypic variable.

The program is able to deal with any combination of phenotypic and genetic data, including situations where only phenotypic or only genetic data are available and situations when each individual is observed through its own combination of phenotypic and genetic markers. See Fig. 5 p. 16 for an overview of the full model.

1.4.3   Output files and graphics

The output of Geneland consists of

- an estimation of the number of HWLE populations,

- a map of the geographic spread of these various populations,

---

[1] available from `http://cran.r-project.org`
[2] `http://cran.r-project.org/bin/macosx/tools`

- a file giving the estimated population membership of each individual

- a file giving the estimated population membership of pixel of the study domain (the size of the pixel being prescribed by the user).

Optional computing options allow to

- account for null alleles (diploid data only)

- account for spatial coordinates uncertainty

Additional outputs include

- computation of pairwise population F_{ST}

- computation of individual population F_{IS}

2   Models

Three types of quantities are involved:

- the (usually unknown) number of populations K

- the parameters (or hidden variable) coding for population membership (of individuals and pixels)

- the parameters of the genetic model conditionally on the the number of populations and on population memberships.

They are modelled separately. K is assumed to follow a uniform distribution between 0 and an upper bound K_{max} prescribed by the user. The genetic and the spatial model are specified conditionally on K. This is described below.

2.1   Mixture models for genetic data

2.1.1   Diploid data: looking for within group Hardy-Weinberg and linkage equilibrium

It is assumed that the overall dataset consists of individuals belonging to K populations, each of these populations being at Hardy-Weinberg equilibrium with linkage equilibrium between loci. For n individuals genotyped at L loci, denoting by f_{klj} the frequency of allele j of locus l in population k, by p_i population membership of individual i ($p_i \in \{1, ..., K\}$) and by $z_{il} = (\alpha_{il}, \beta_{il})$ the genotype of individual i at loci l, HWLE writes:

$$\pi(z) = \pi((z_{il})_{il}) = \prod_{i=1}^{n} \prod_{l=1}^{L} f_{p_i l \alpha_i} f_{p_i l \beta_i} (2 - \delta_{\alpha_i}^{\beta_i}) \tag{1}$$

where $\delta_{\alpha_i}^{\beta_i} = 1$ if $\alpha_i = \beta_i$ and 0 otherwise.

2.1.2   Haploid data: multinomial distribution and linkage equilibrium

For haploid data, the model assumes a multinomial distribution of genotypes conditionally on allele frequencies and population memmberships and linkage equilibirum is also assumed.

2.1.3   The uncorrelated frequency model

Allele frequencies in the various sought populations are unknown and although they are not of direct interest, it is convenient to introduce them in the statistical computations. Indeed, once they are introduced and although they are unknown, Equation (1) allows to compute the likelihood. Plugging this equation in an iterative scheme known as Metropolis-Hastings algorithm allows to start from arbitrary values for all unknown parameters and to modify them in such a way that after many iterations, these values are close to the true values. The trick consisting in including extra unknown parameters in the inference not of direct interest but for computational purpose is known as data augmentation in statistics.

Once we have introduced the allele frequencies in the various sought populations, we need to place a prior distribution on them in view of Bayesian inference (see section B).

For each population and each locus, the entries of the vector $f_{kl1}, ..., f_{klJ_l}$ sum up to one. The simplest probability distribution fulfilling this condition is the Dirichlet distribution [3]. Beyond this algebraic property, it also has the interest to comply with a Wright-Fisher island model, the asymptotic distribution of allele frequency being Dirichlet under this model.

This distribution depends on a single vector parameter which might vary across populations and loci. Assuming that this parameter α_{kl} is not common across populations and loci, assuming Dirichlet a priori distribution for f_{klj} writes:

$$\pi(f_{klj}) = \Gamma(J_l) \tag{2}$$

This probability does not depend on the actual values taken by f_{klj} and this model turns out to give the same a priori probability to any allele frequencies. The key assumption consists now in assuming that the vectors $f_{kl.}$ are mutually independent across populations. Independence of the vectors $f_{kl.}$ is of course assumed across loci.

2.1.4 The correlated frequency model

The previous model is somehow over simplistic in the sense that most often, allele frequencies tend to be similar in different populations, (e.g. rare alleles in a certain populations are also rare in other populations). From a statistical point of view, this property can be viewed as a correlation of f_{klj} and $f_{k'lj}$ for some populations k and k'. This correlation can be viewed as a simple but useful summary of the common recent (micro-)evolutionary history of populations k and k'.

The correlated frequencies model has been introduced previously (see e.g. [Balding, 2003],[Nicholson et al., 2002]) to account exactly for this property.

In this model, one introduces the frequencies of an ancestral population denoted by f_{Alj} also assumed to be independently Dirichlet distributed and a vector of population specific drift parameters $(d_1, ..., d_K)$ so that $f_{kl.}|f_A, d$ has a Dirichlet distribution

$$D\left(f_{Al1}(1 - d_k)/d_k, ..., f_{AlJ_l}(1 - d_k)/d_k\right) \tag{3}$$

In this model and conditionally on f_A and d, the frequencies are independent across populations, but marginally (integrating out f_A and d) elementary computations (see [Guillot, 2008]) show that the correlation of allele frequencies across population is:

$$Cor(f_{klj}, f_{k'lj}) = 1 \left/ \left[1 + E[d_k] \frac{E[f_{Alj}] - E[f_{Alj}^2]}{E[f_{Alj}^2] - E[f_{Alj}]^2} \right] \right. \tag{4}$$

In the most general case, the distribution of the f_{klj}s in the uncorrelated model may depend on population-, locus- and allele-specific parameters α_{klj}. In practice, the α_{klj} are always assumed to be common across populations, locus and alleles, and most often set to one. Similarly, in the correlated model, the f_{Alj} might have locus- and allele-specific parameters but I do not consider this case here. I set it to one as it is most often done in practice, although the effect of this assumption has not been yet thoroughly assessed in the context of clustering (but see [Foll et al., 2008] in another context). Independence is always assumed across loci.

Specifying fully the model also requires to place a prior on the drift parameters d_k. As this parameter has to lie in $[0, 1]$, it is natural to consider a Beta prior, with independence across populations. A Beta distribution depends on two parameters.

The correlated model can be viewed as a Bayesian and biologically grounded way to make inference under the uncorrelated model with population-, locus- and allele-specific parameters.

It has been observed that using the correlated frequency model could be more powerful at detecting subtle differentiations. On the other hand, this model seems to be more prone to algorithm instabilities and more sensitive to departure from model assumptions (e.g. presence of isolation-by-distance). It is recommended to start data analysis with the uncorrelated frequency model and then to check how these initial results are modified by the use of the correlated frequency model. It is also important to check ex-post that the inferred groups are significantly differentiated and at HWLE.

---

[3]See en.wikipedia.org/wiki/Dirichlet_distribution.

2.2 Mixture model for phenotypic data

We assume here that we have a data-set consisting of n individuals sampled at sites $\mathbf{s} = (s_i)_{i=1,\dots,n}$ (where s_i is the two-dimensional spatial coordinate of individual i), observed at some phenotypic variables denoted $\mathbf{y} = (y_{ij})_{\substack{i=1,\dots,n \\ j=1,\dots,q}}$. Our approach also encompasses the case where sampling locations are missing (or considered to be irrelevant). The only constraint that we impose is that if spatial coordinates are used, they must be available for all individuals.

As above, we assume that each individual sampled belongs to one of K putative clusters and that variation in the data can be captured by cluster-specific (location and scale) parameters.

Denoting by p_i the cluster membership of individual i ($p_i \in \{1, ..., K\}$), we assume that conditionally on $p_i = k$, y_{ij} is drawn from a parametric distribution with cluster-specific parameters. Independence is assumed within and across clusters conditionally on cluster membership. This means in particular that there is no residual dependence between variables not captured by cluster memberships. We assume that the y values arise from a normal distribution. Each cluster is therefore characterized by a mean μ_{kj} and a variance σ_{kj}^2 and our model is a mixture of multivariate independent normal distributions. We use the natural conjugate prior family on $(\mu_{kj}, 1/\sigma_{kj}^2)$ for each cluster k and variable j. Namely, we assume that the precision $1/\sigma_{kj}^2$ (i.e. inverse variance) follows a Gamma distribution $\mathcal{G}(\alpha, \beta)$ (α shape, β rate parameter) and that conditionally on σ_{kj}, the mean μ_{kj} has a normal distribution with mean ξ and variance σ_{kj}^2/κ. In the specification above, α, β, ξ and κ are hyper-parameters. Details about their choice are discussed in Guillot et al. [2012].

2.3 Models underlying population membership

2.3.1 The non-spatial model

I denote by p a vector parameterising the population memberships. In case population membership is modeled at the individual level, this vector can be simply $p = (c_1, ..., c_n)$ where $c_i \in \{1, ..., K\}$; In this case, the simplest form of prior that can be placed on it is an i.i.d prior $\pi(p|K) = 1/K^n$.

Spatial representation for six simulations from this prior model are given on figure 1.
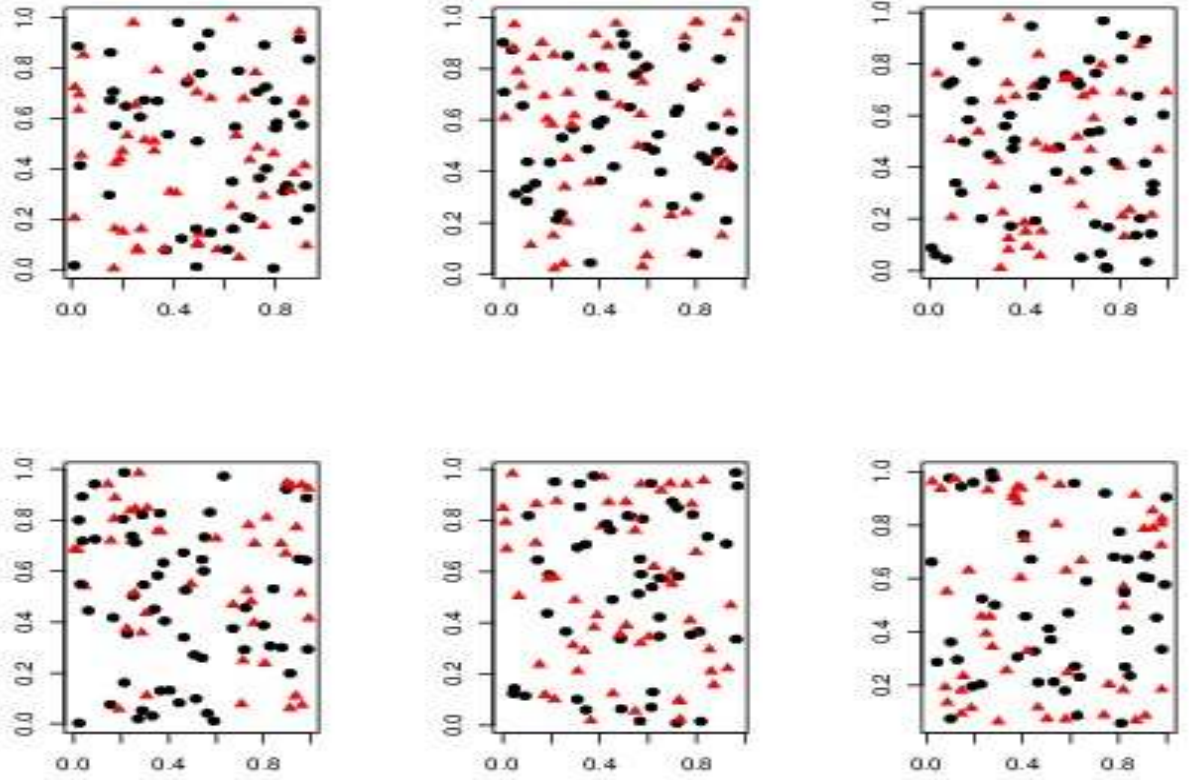
Figure 1: Six examples of 100 individuals belonging to two populations spread totally at random across space.

The previous model is numerically convenient but it is biologically questionable in the sense that one may wonder how differentiation might have occurred between populations in case of such spatial overlap between them.

2.3.2 The spatial model

This model consists in assuming that spatial domain of each population can be approximated by the union of a few polygonal domains, see figure 2 for examples.

This model corresponds to the spatial patterns that can be expected when differentiation occurs by limited gene flow induced by the presence of physical barriers such as road, rivers, mountain ranges, human activity.

Formally, the colored Poisson-Voronoi tessellation model consists in assuming that there is an unknown number of polygons m that approximate the true pattern of population spread across space. These polygons are "centred" (this term is actually a bit inaccurate mathematically, but see below) around spatial points $u_1, ..., u_m$ and each polygon belongs to one of the K population which is coded by an integer (or color for graphical representation) denoted by $c_1, ..., c_m$. Examples are given on figure 4.

The exact mathematical definition of the colored Poisson-Voronoi tessellation model is as follows:

- the number of polygons follows a Poisson distribution with parameter λ: $m \sim \text{Poisson}(\lambda)$

Figure 2: Six examples of 100 individuals belonging to two populations where the spatial domain of each population can be approximated by the union of a few polygonal domains.

- conditionally on m, there are m mutually independent points $u_1, ..., u_m$ with uniform distribution over the study domain D: $(u_1, ..., u_m)|m \overset{\text{i.i.d}}{\sim} \text{Uniform}(D)$

- each points u_i defines a set V_i of points in D that are closer to u_i than to any other points in $(u_1, ..., u_m)$. This set V_i is the i-th cell of the so-called Voronoi tessellation induced by $(u_1, ..., u_m)$.

- the points $(u_1, ..., u_m)$ (or now equivalently the sets $V_1, ..., V_m$) receive a mark with value in $\{1, ..., K\}$ coding for population membership and displayed as different colors. These colors a assumed to be sampled from a probability distribution, which in GENELAND (but this is not a requirement) is assumed to be an independent, identically distributed and uniform distribution on $\{1, ..., K\}$: $(c_1, ..., c_m)|m \overset{\text{i.i.d}}{\sim} \text{Uniform}(\{1, ..., K\})$

2.4 Additional modelling options

2.4.1 Null alleles

A well known source of potential problems is the presence of null alleles arising from variation in the nucleotide sequences of flanking regions that prevent the primer annealing to template DNA during PCR amplification of the microsatellite locus [Dakin and Avise, 2004, Pompanon et al., 2005]. The presence of null alleles results in an excess of homozygous genotypes within a population as compared to the expected proportion under Hardy Weinberg Equilibrium (HWE) and Linkage Equilibrium (LE) [Callen et al., 1993, Paetkau et al., 1995] while the model in GENELAND is based on HWE and LE within the sought clusters.

In GENELAND, the putative presence of null allele(s) can be explicitly taken into account for diploid data through an optional computing step. When this option is used, genotype ambiguity (homozigotes) is accounted for
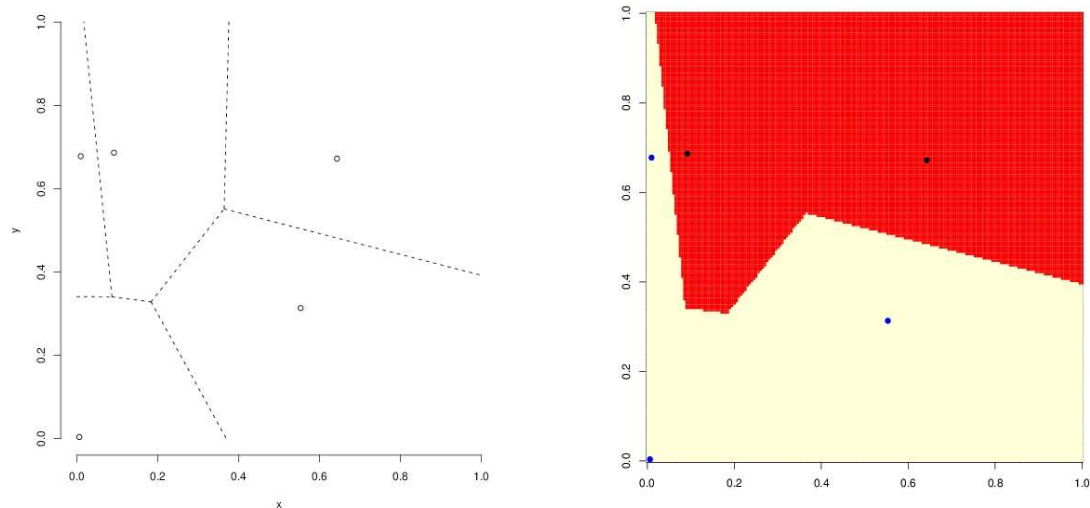
Figure 3: Example of colored Poisson-Voronoi tessellation. Left panel: location of cell "centre" and Voronoi cells induced. Right panel: a example of colored tessellation obtained after coloring at random each cell as red or white. In this example, the number of population $K = 2$ and the number of polygons $m = 5$. There is no attempt to represent the location of any sampled individual.

and null alleles frequencies is estimated along the clustering algorithm. With the option `filter.NA=TRUE` in function `MCMC`, null allele frequencies at each locus are estimated. They can be viewed with function `EstimateFreqNA`. This is also available through the GUI.

Note that if this option is used in the default setting, all double missing genotypes will be interpreted as double null alleles. This can induce an over-estimation of null allele frequencies if some of the missing data are not null alleles (i.e. not due to amplification problems in the PCR), for example if some loci are missing for all individuals of certain sampling units. In order to minimize this kind of problems, an optional argument to function `MCMC` named `miss.loc` can be passed. The variable passed must be an $n \times L$ matrix (nb indiv. $\times$ nb loci). The entry of this matrix for line i and column l must be set to 1 if there was no attempt to record the genotype of indiv. i at locus l ("genuine missing locus") and to 0 otherwise (if data are available or are missing because the PCR failed). If no variable is passed to `miss.loc`, a matrix with only 0s is passed by default and all double missing values will be treated as null alleles.

2.4.2 Coordinates uncertainty

You may want to treat the spatial coordinates as uncertain for any or a combination of the following reasons:

1. The individuals under study are non mobile (plants, animals species with very limited vagility as compared to the scale of the study domain) but the coordinates have been recorded with an error or they have been recorded with a limited precision only (e.g. each individual has been affected to a small administrative unit).

2. The individuals under study are normally non mobile but a displacement might have been induced by the observation (e.g. hounding by hunters).

3. The individuals under study are mobile and they have a home range whose characteristic scale is non negligible as compared to the size of the study domain.

4. Even if none of the previous conditions holds, but if your dataset has samples sharing the same coordinates, then allowing some uncertainty in the coordinates will allow to have samples with the same coordinates to be assigned to different populations. It can be useful to detect migrants.

Under conditions (1) and (2), the notion of "true" coordinates makes sense and refers to the location where each individual normally lives, but these locations have not been observed. Under condition (3), there is no "true
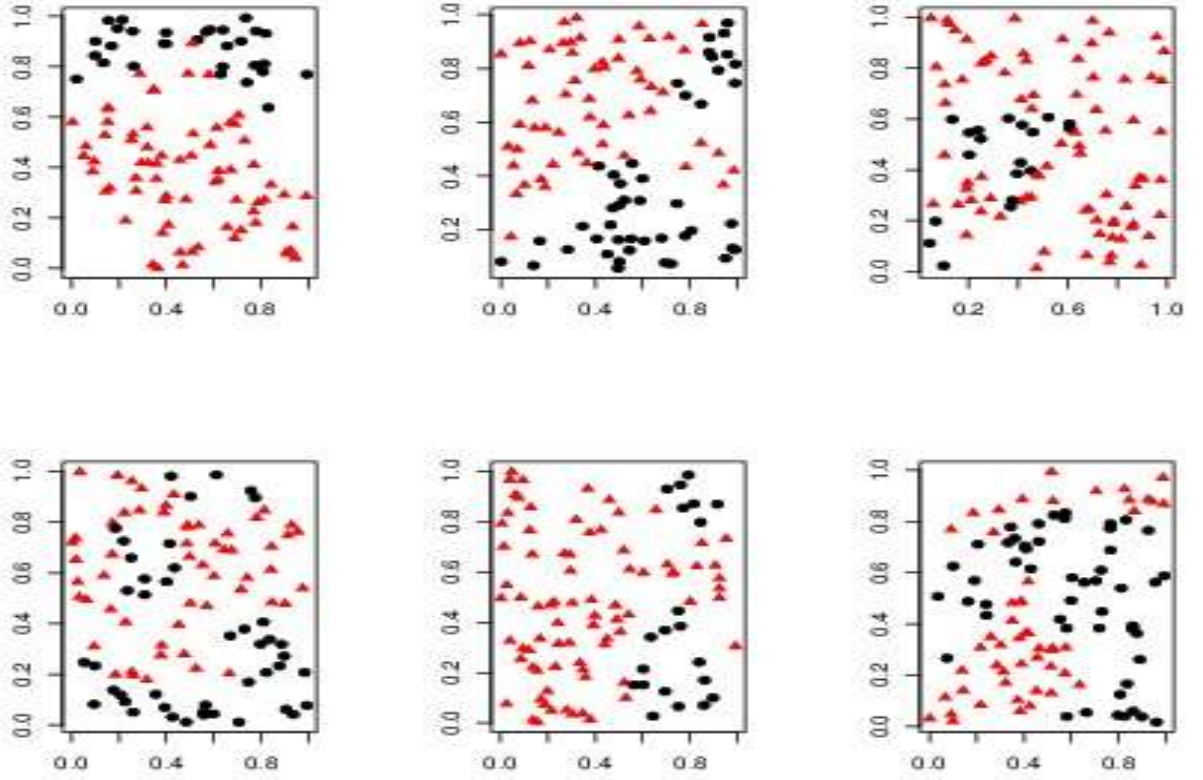
Population membership
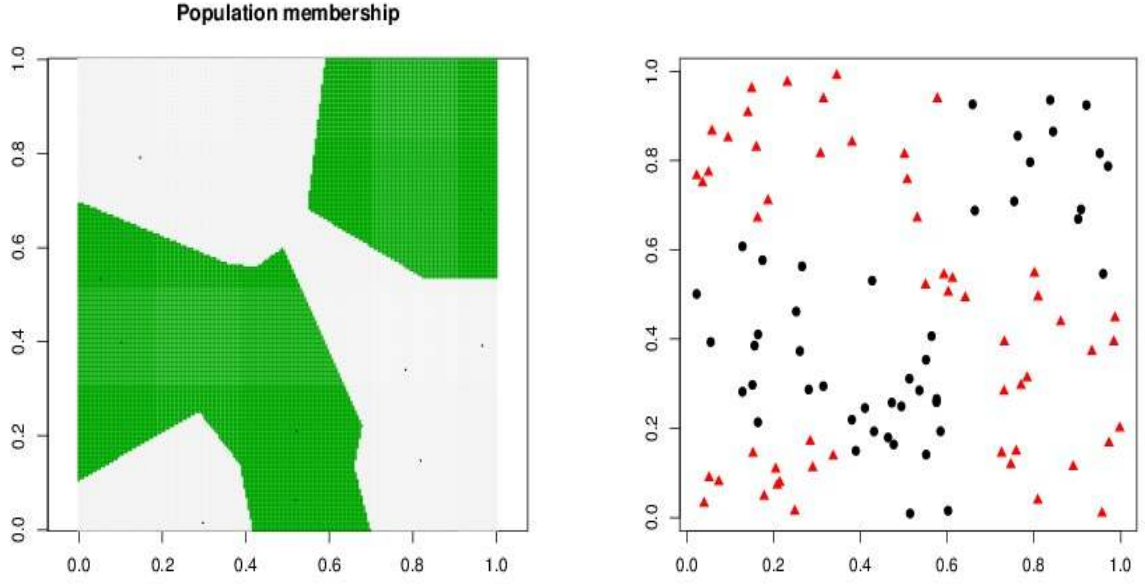
Figure 4: An example of 100 individuals belonging to two populations where the spatial domain of each population can be approximated by the union of a few polygonal domains. Left: spatial spread of individuals, right: corresponding polygons. In this example, $n = 100$, $K = 2$ and the number of polygons $m = 10$. The points u_i are depicted as tiny black dots on the left hand side panel. They are not depicted on the right hand side for clarity.

coordinates", i.e. there is no particular location where each individual could be considered to live with certainty. In any of the previous conditions, the use of the recorded coordinates as locations where the individuals live is inaccurate and can be misleading in the inferences. It is recommended in this case to treat the observed coordinates as uncertain. The way GENELAND does it is to consider that the observed coordinate of each individual is the sum of the true coordinate and of a random noise of small magnitude. This random noise can be interpreted as the movement induced by capture in case of hounding (condition b), in the other cases it says that an individual has been observed somewhere but that it could have been observed anywhere around as well.

There is a parameter prescribing the amount of uncertainty attached to spatial coordinates (parameter `delta.coord` in function MCMC).

If this parameter is set to 0, spatial coordinates are considered as true coordinates, if this parameter is set to a value `delta.coord`>0, it is assumed that observed coordinates are true coordinates blurred by an additive noise uniform on a square of side `delta.coord` centered on 0. It is assumed that this parameter is given in the same units as the spatial coordinates. See also section A.21.
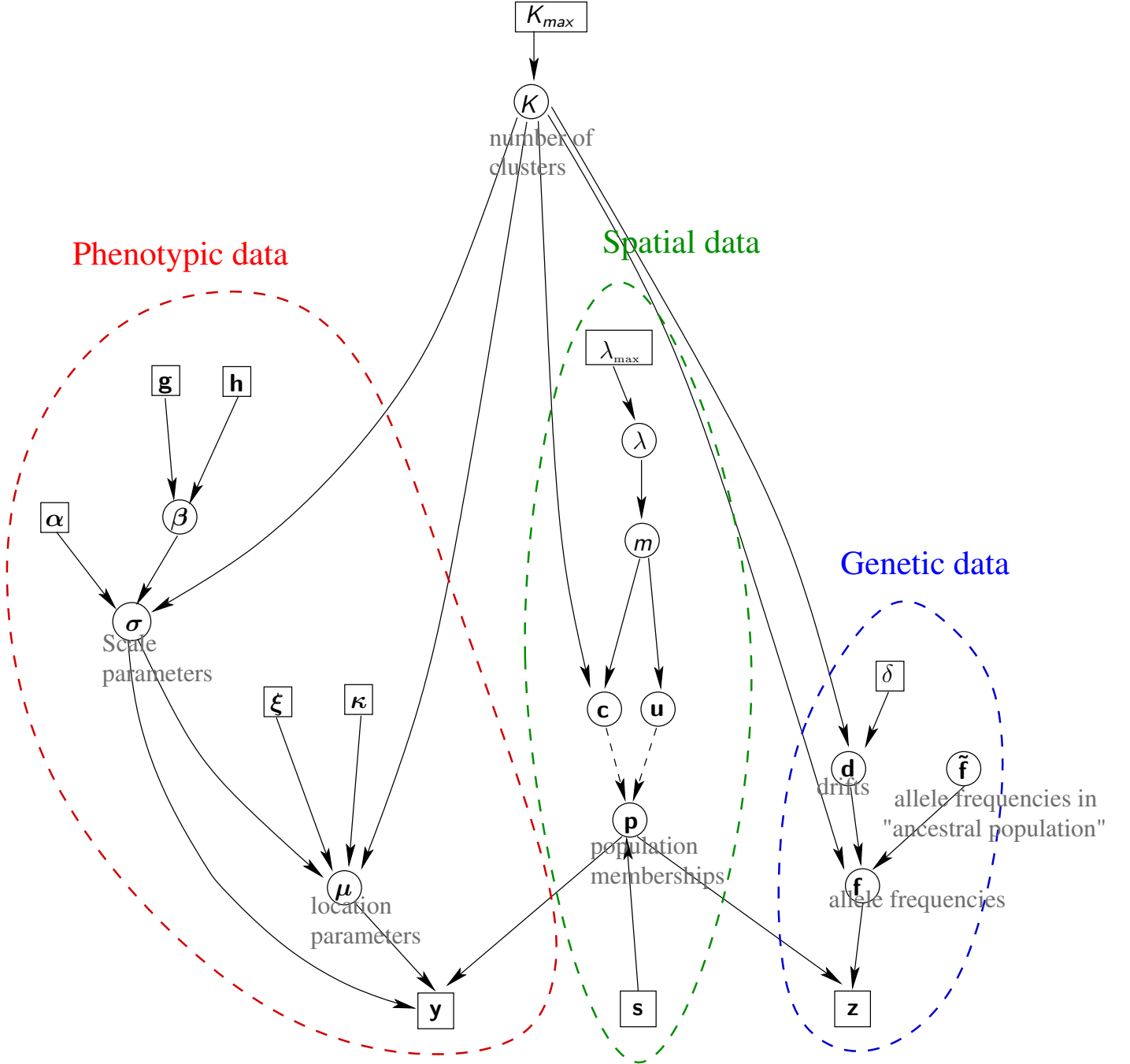
Figure 5: Graph of the global model. Continuous black lines represent stochastic dependencies, dashed black lines represent deterministic dependencies. Boxes enclose data or fixed hyper-parameters, circles enclose inferred parameters. Bold symbols refer to vector parameters. The red, green and blue dashed lines enclose parameters relative to the phenotypic, geographic and genetic parts of the model respectively. The parameters of interest to biologists are the number of clusters K, the vector $\mathbf{p}$ which encode the cluster memberships, and possibly allele frequencies $\mathbf{f}$, mean phenotypic values $\boldsymbol{\mu}$, phenotypic variance $\boldsymbol{\sigma^2}$ which quantify the genetic and phenotypic divergence between and within clusters. Other parameters can be viewed mostly as nuisance parameters.

2.5   Admixture model and hybrid zones

2.5.1   Model

In the models described in sections 2.1-2.3, each individual is assumed to have ancestries in one cluster only. This assumption might not be relevant, especially if the population under study undergone recent admixture events. Versions of GENELAND $\geq 3.3.0$ include a model where individuals have mixed ancestries [Guedj and Guillot, 2011]. The likelihood of this model is similar to that of STRUCTURE [Pritchard et al., 2000]. Introducing the matrix $q = (q_{ik})$, where q_{ik} refers to individual i's genome proportion originating in cluster k, for diploid genotypes we have:

$$L(z_{il}|f, q) = \sum_{k=1}^{K} q_{ik} f_{klz_{il1}} f_{klz_{il2}} (2 - \delta_{z_{il1}}^{z_{il2}}) \,, \tag{5}$$

and for haploid data we have

$$L(z_{il}|f, q) = \sum_{k=1}^{K} q_{ik} f_{klz_{il}} \,. \tag{6}$$

The model introduced now differs from earlier versions of GENELAND in that it models admixture and from STRUCTURE in that it is spatial. Those two features are accounted for as follows: each vector of admixture proportions $q_{i.} = (q_{ik})_{k=1,...,K}$ is assumed to follow a Dirichlet distribution $\mathcal{D}(\alpha_{i1}, ..., \alpha_{iK})$. We denote by d_{ik} the distance of individual i to cluster k (in particular, $d_{ik} = 0$ if individual i has been sampled in cluster k) and we assume a deterministic relationship

$$\alpha_{ik} = a \exp(-d_{ik}/b) \tag{7}$$

By a standard property of the Dirichlet distribution, under equation (7) the expected value of q_{ik} is

$$E[q_{ik}] = \frac{e^{-d_{ik}/b}}{\sum_k e^{-d_{ik}/b}} \tag{8}$$

In presence of $K = 2$ clusters in contact along a hybrid zone, and if individual i belongs to cluster 1, then by definition $d_{i1} = 0$ and we get

$$\begin{aligned} E[q_{i1}] &= \frac{e^{-d_{i1}/b}}{e^{-d_{i1}/b} + e^{-d_{i2}/b}} \\ &= \frac{1}{1 + e^{-d_{i2}/b}} \end{aligned} \tag{9}$$

i.e. the well known sigmoid function familiar to people studying hybrid zones. Under this model, the width of the cline (defined as the inverse of the maximum gradient) is $w = 1/4b$. The variation of the expected admixture coefficients is illustrated in figure 6.

Parameter a is a-dimensional, it does not affect the expected value of q_{ik} but controls its variance with $V[q_{ik}] \propto 1/a$. Large a values correspond to datasets with individuals displaying pretty similar admixture proportions within clusters. Parameter b is a spatial scale parameter, it has the dimension of a distance and is expressed in the same unit as spatial coordinates. Large b values correspond to situations where admixture coefficients are loosely structured in space. At the limit where $b = +\infty$, the vector $q_{i.}$ follows a flat Dirichlet distribution and the model does not display spatial features at all. Conversely, at the limit value $b = 0$, all individuals display admixture proportions that are 0 or 1 with a spatial pattern mirroring exactly the underlying Poisson-Voronoi tessellation. We place a uniform prior on a and b and assume independence of these two parameters.

2.5.2   Inference procedure

An exact Bayesian inference would estimate all parameters of the admixture model by joint MCMC simulation of (q, a, b) together with any other parameters involved in the model (number of gene pools, tessellation parameters and allele frequencies). We believe that the implementation of this strategy would offer a number of challenges, in particular when the number of clusters is unknown. For this reason, we suggest an alternative approximate two-stage strategy:
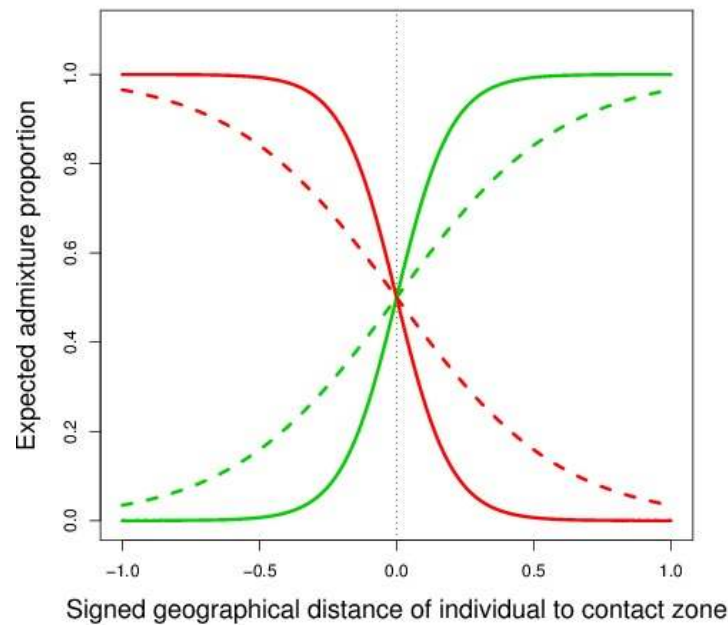
Figure 6: Examples of spatial variation of expected admixture proportions in presence of two clusters. Individuals whose proportions are displayed here are assumed to be continuously located along a linear transect crossing perpendicularly a hybrid zone. Red line: expected admixture proportion q_{i1}, green line: expected admixture proportion q_{i2}. Continuous lines: $a = 1$, $b = 0.1$, dashed lines: $a = 1$, $b = 0.3$. Note that the curves are exactly sigmoid (logistic) functions.

1. estimate allele frequencies and cluster locations under the non-admixture model. Denoting by $\theta_{no\ adm.}$ the vector of parameters in the no-admixture model, this amounts to sampling from $\pi(\theta_{no\ adm.}|z)$. This produces an estimate $\hat{\theta}_{no\ adm.}$, in particular it provides an estimate of the location of the putative hybrid zone.

2. estimate (q, a, b) by MCMC simulation from the distribution of (q, a, b) conditionned by the data and the parameters estimates obtained from the non-admixture Geneland run. This amounts to sampling from $\pi((q, a, b)|\hat{\theta}_{no\ adm.}, z)$.

With the command-line, the first step is implemented in function `MCMC`, the second step in function `HZ`. Estimates of parameters of the hybrid zone model can be visualized with function `show.estimate.hz`.

18
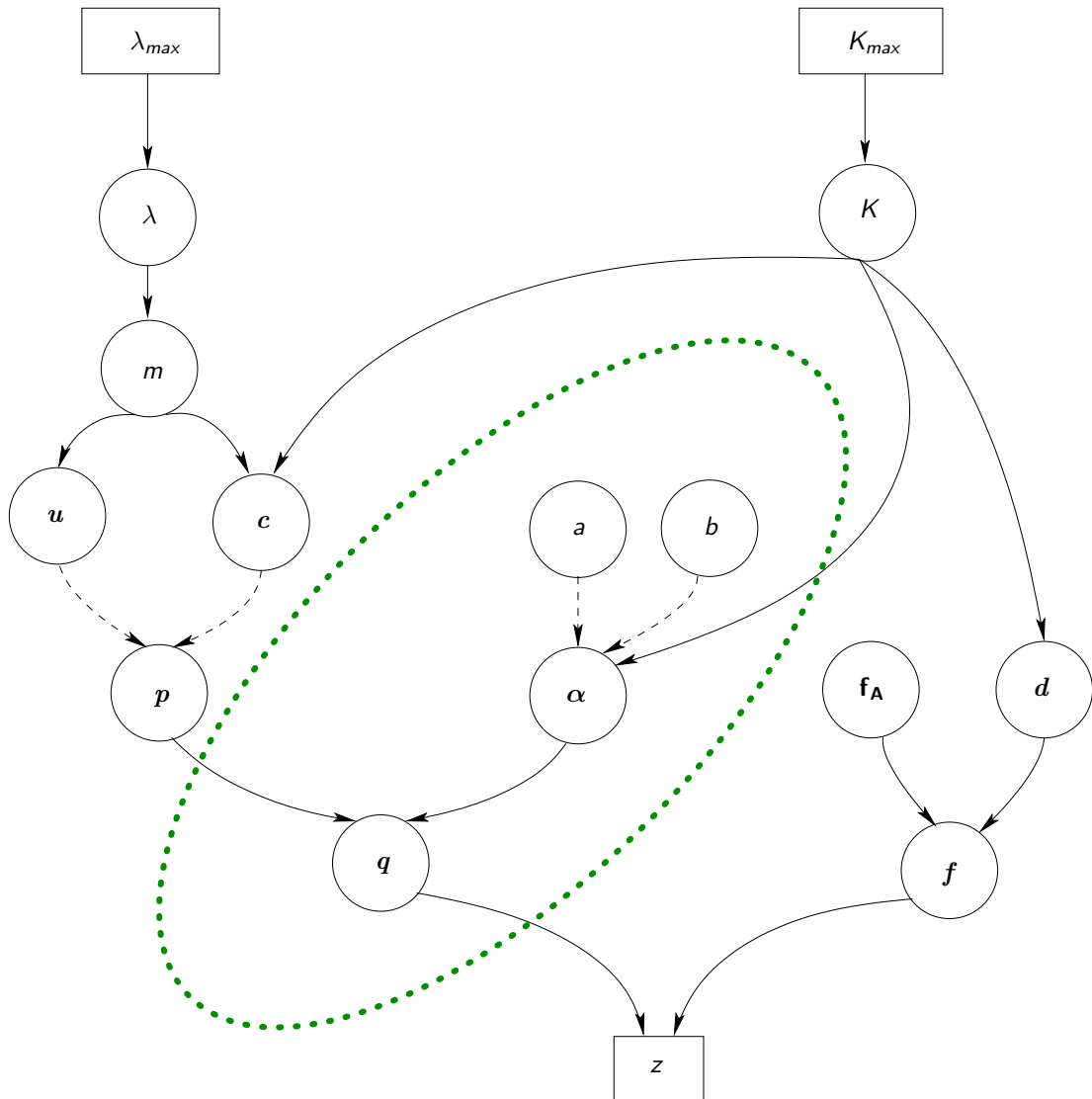
Figure 7: Directed acyclic graph of the genetic model(s) implemented in GENELAND. Continuous lines represent stochastic dependencies, dashed lines represent deterministic dependencies. Squared boxes enclose data or fixed hyper-parameters, rounded boxes enclose inferred parameters. The thick green dotted line enclose parameters of the admixture model.

3   Data format

3.1   Genotypes file

Assuming that you have data for n individuals genotyped at L loci, the data must be arranged in

- a plain ascii file

- without any extra invisible characters (like in MS-Word .doc files)

- with one line per individual

- each allele must be coded by an integer

- the number of digits of each field is arbitrary and can vary across loci

- extra header lines are not allowed when data are loaded via the GUI. Headers lines can be dealt with via the R function `read.table`. The way these lines are handled when the data are loaded is prescribed through the arguments of .

- missing data are allowed. Users of the GUI have to specify the character string coding missing value in the `missing data symbol` sub-menu of the main menu. For users of the R command-line interface loading their data with the R function `read.table`, missing values can be coded in the raw text data file by any arbitrary character string, (e.g. `000`, `00`, `NA` or `-999`). The character string used to encode missing data must be specified through the argument `na.strings` of function `read.table`. By default, when data are loaded through `read.table`, it is assumed that missing data are coded as `NA`.

- for haploid organisms with L loci, the genotype file must have L columns.

3.1.1   Diploid codominant data file

This applies to microsatellites or SNPs data for diploid organisms. The file should have one line and $2 \times L$ columns per individual For $L = 10$, the two first lines of the genotype file should look like:

```
198 000 358 3)62 141 141 179 000 208 224 243 243 278 284 86 88 120 124 238 244
200 202 000 358 141 141 183 183 218 224 237 243 276 278 88 88 120 124 240 244
.     .     .     .     .     .     .     .     .     .     .     .     .     .     .     .     .     .
.     .     .     .     .     .     .     .     .     .     .     .     .     .     .     .     .     .
.     .     .     .     .     .     .     .     .     .     .     .     .     .     .     .     .     .
```

In the example above, missing values are coded as 000. In the command line version, this file corresponds to argument `geno.dip.codom` of function `MCMC`.

3.1.2   Diploid dominant data file

This applies to AFLP data for diploid organisms.

The file should have one line and L columns per individual. Absence or presence of the dominant allele should be coded as 0/1/ For $L = 10$, the two first lines of the genotype file should look like:

```
0 1 1 1 0 1 0) 0 0
1 0 0 1 1 0 0 1 000
. . . . . . . . .
. . . . . . . . .
. . . . . . . . .
```

In the example above, missing values are coded as 000. In the command line version, this file corresponds to argument `geno.dip.dom` of function `MCMC`.

Diploid codominant data and diploid dominant data can be analyzed jointly. Diploid dominant data and haploid data can not be analyzed jointly.

3.1.3   Haploid data file

This applies to microsatellites or SNPs data for haploid organisms or to mitochondrial DNA data.

The file should have one line and L columns per individual For $L = 10$, the two first lines of the genotype file should look like:

```
198 000 358 3)62 141 141 179 000 208 224
200 202 000 358 141 141 183 183 218 224
  .   .   .   .   .   .   .   .   .   .
  .   .   .   .   .   .   .   .   .   .
  .   .   .   .   .   .   .   .   .   .
```

In the example above, missing values are coded as 000. In the command line version, this file corresponds to argument `geno.hap` of function `MCMC`.

Diploid codominant data and haploid data can be analyzed jointly. Diploid dominant data and haploid data can not be analyzed jointly.

3.2   Coordinates file

3.2.1   File format

If you want to use the spatial model, you should have a file containing spatial coordinates of the individuals sampled. The coordinate file must be such that

- there is one line per individual and two columns (x-axis and y-axis coordinate),

- the units do not matter

- it is assumed that the coordinates are planar coordinates such as UTM coordinates

- whether your coordinates are genuine planar coordinates or spherical coordinates does not appear in the file format so that you can actually also input coordinates given as spherical coordinates. They will be interpreted as planar coordinates by the GENELAND functions. The quality of this approximation varies from very good (small domain, close to the equator) to very bad (large domain, close to a pole).

- extra header lines are not allowed when data are loaded via the GUI. Headers lines can be dealt with via the R function `read.table`. The way these lines are handled when the data are loaded is prescribed through the arguments of .

- missing data are not allowed in this file.

- if some coordinates are missing, you can either substitute an estimated value or do the analysis with GENELAND without spatial coordinates at all using the non spatial model.

The two first lines of a coordinate file should look like:

```
25.6 745.2)
54.1 827.8
  .    .
  .    .
  .    .
  .    .
```

3.2.2   Pre-processing coordinates to convert Lon/Lat into UTM

This conversion can be done by function `convUL` of the R package `PBSmapping` or by function `mapproject` of the R package `mapproj`. To use function `convUL`, your matrix of coordinates must have two columns named ''X'' and ''Y'', a projection attriubute (''LL'' or ''UTM'') and a zone attribute (a number between 1 and 60). For example, if your coordinate matrix is named `coord` and is in Lont/Lat format, it can be refromatted in UTM by:

```
colnames(coord) <- c("X","Y")
attr(coord, "projection") <- "LL"
attr(coord, "zone") <- NA
coord.utm <- convUL(coord)
```

See ?   convUL for detail.

4   Examples of data analysis using the graphical user interface (GUI)

4.1   Launching the GUI

- launch R

- load GENELAND by typing `library(Geneland)` in the R prompt

- launch the GENELAND GUI by typing `Geneland.GUI()` in the R prompt.

This will open open a window as below:

23
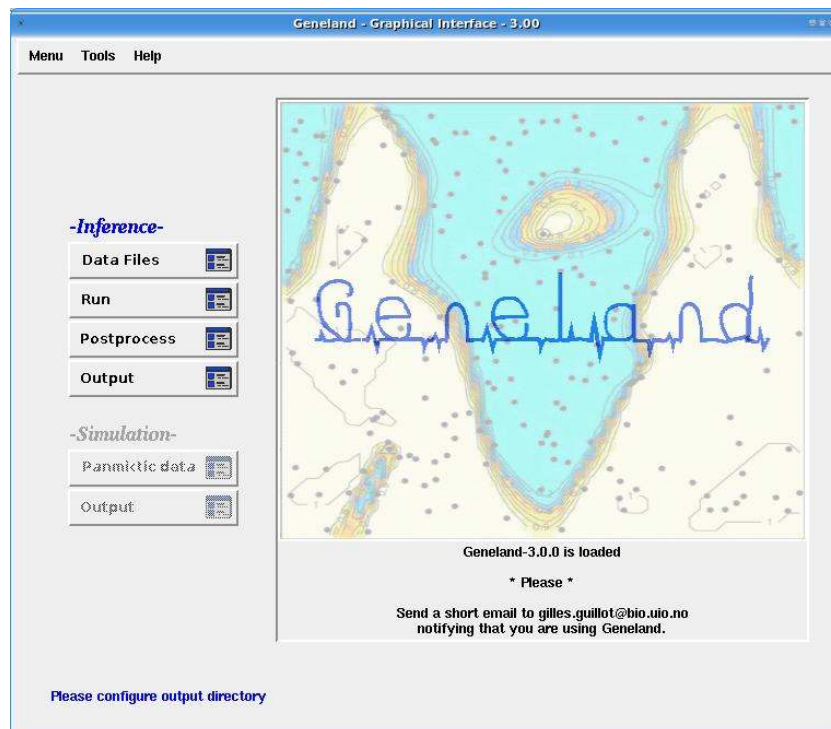
4.2   Selecting input and output files

Select the fields separator in your data files:
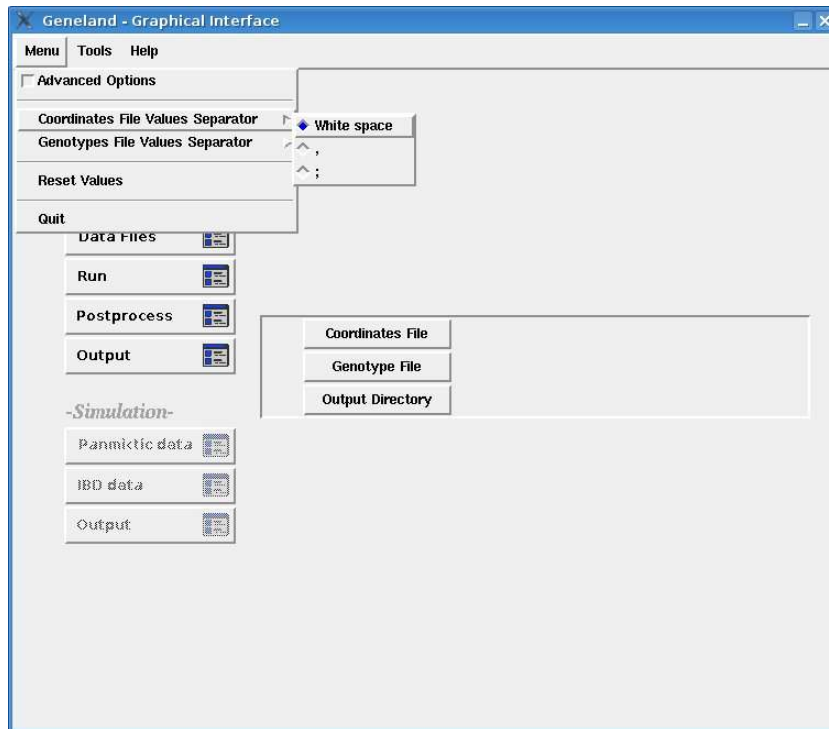
Select your data files and the directory containing your output files
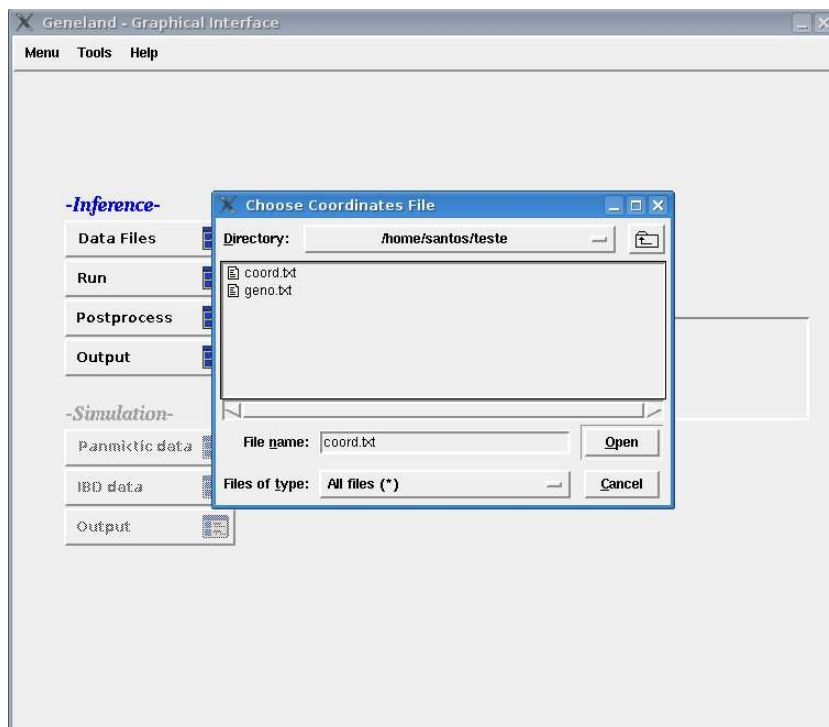
Important note: the path can not be longer than 255 characters. You can avoid using long paths by selecting the working directory (under Windows) or by launching R in a shell from a particular directory (under Linux).

4.3   Inference

We now carry out an analysis to infer the number of populations and their spatial boundaries for this dataset. This can be performed by selecting MCMC simulation parameters as follows:
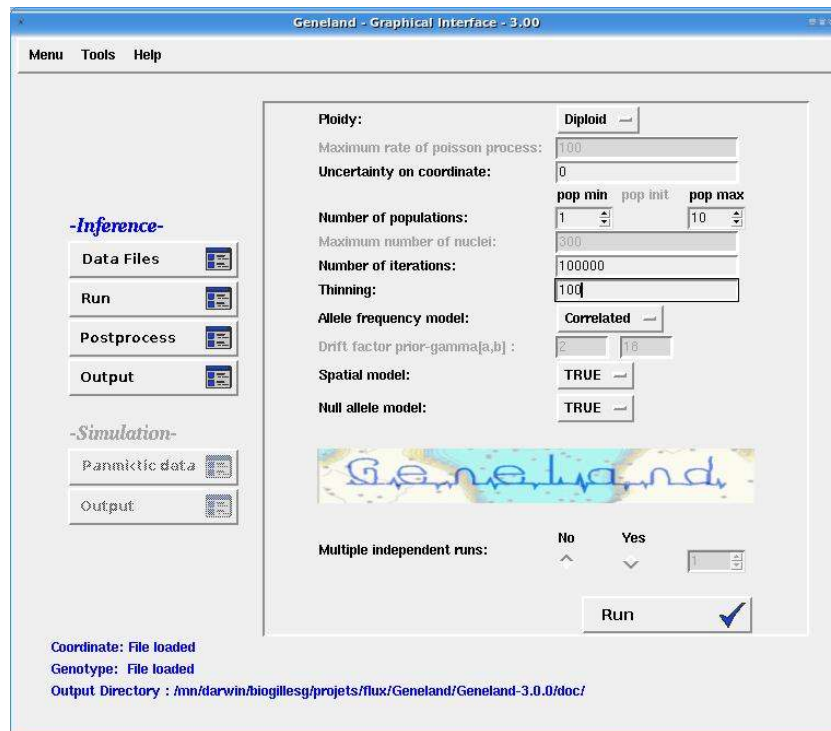
In the above example we select

- diploid organism

- no uncertainty attached to spatial coordinates

- the number of HWLE populations is unknown and hence treated as simulated variable along the MCMC simulations allowed to vary between 1 and 10

- the number of MCMC iterations will be 100000 (`nit=100000`)

- and only each 100th iteration will be saved on the disk (in total, 1000 iterations will be saved)

- combined with the correlated frequency model (`freq.model="Correlated"`).

- using the spatial model (`spatial TRUE`),

- in the current working directory (`path.mcmc="./"`).

4.4   Post-processing MCMC outputs

The call to function `MCMC` generates different files in the directory specified by the argument `path.mcmc`. Information is extracted from this file through a call to function `Post.Process.Chain` as follows:
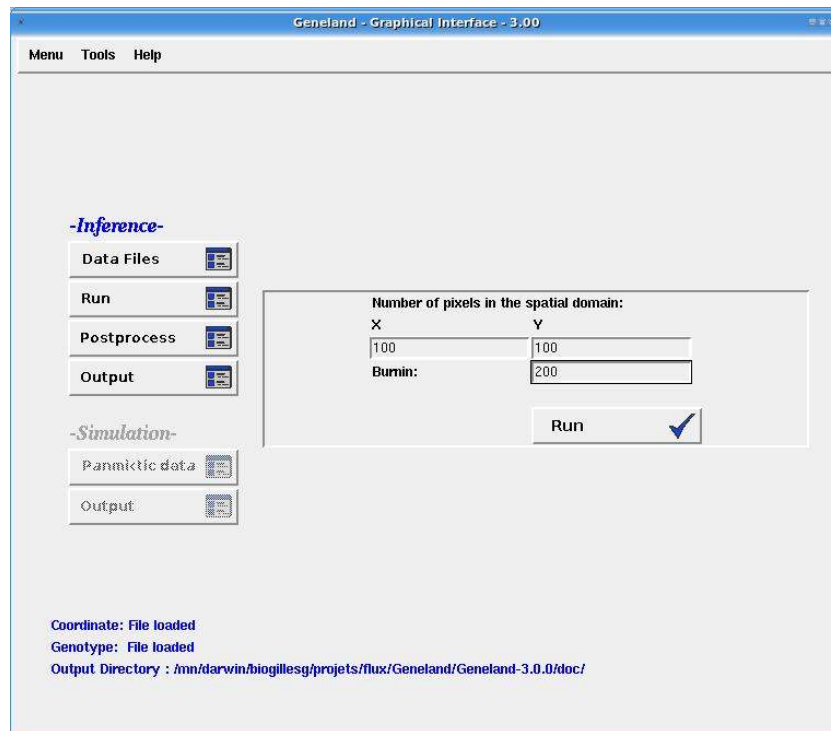
This will extract information from MCMC simulation with

- an horizontal discretization of the study domain in 100 pixels (`nxdom=100`)

- a vertical discretization of the study domain in 100 pixels (`nydom=100`)

- a burn-in of 200 saved iterations, i.e. discarding the 200 first saved iterations (`burnin=200`)

The parameters `nxdom` and `nydom` are only graphical parameters to set the resolution of the final maps. Here are some recomandations to set them.

- Set `nxdom` and `nydom` so as to avoid having two sampling sites in the same pixel.

- Set `nxdom` and `nydom` so as to have the same resolution on both axis. For example, if you have a study domain of 300 km × 200 km, make sure that `nxdom` / `nydom` ≈ 3/2.

- Avoid large values (larger than say 500). It would increase the resolution of the map in a way that is not visually detectable and it might saturate the virtual memory of your computer.

- The post-processing step is independent of the Markov chain computations and you can do as many tries as you want..

- Why not having included the post-processing step as an automatic step of function MCMC? Because one may want to play with the burn-in and the resolution without re-launching the whole MCMC computations.

4.5   Generating graphical and numerical outputs

Clik on menu output:
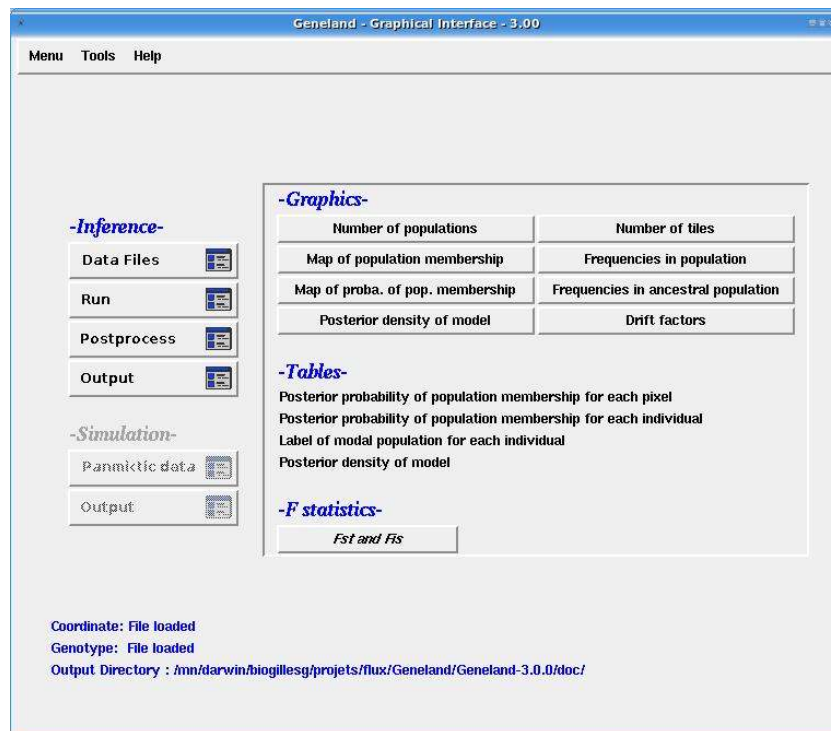
4.5.1   Estimated number of HWLE populations

The number of population simulated from the posterior distribution can be visualised by clicking on `Number of populations`

   This produces a plot like:
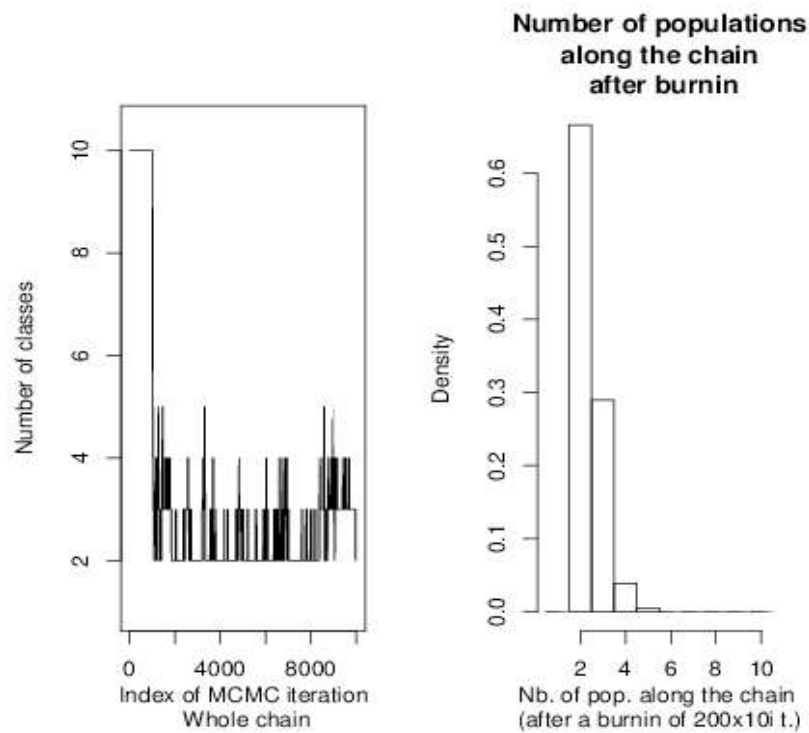
This run displays a clear mode at $K = 2$ which is hence the *maximum a posteriori* estimate of K.

4.5.2   Map of estimated population membership

A map of estimated population membership (by posterior mode) can be obtained by clicking `Map of population membership`
   This produces a plot like:
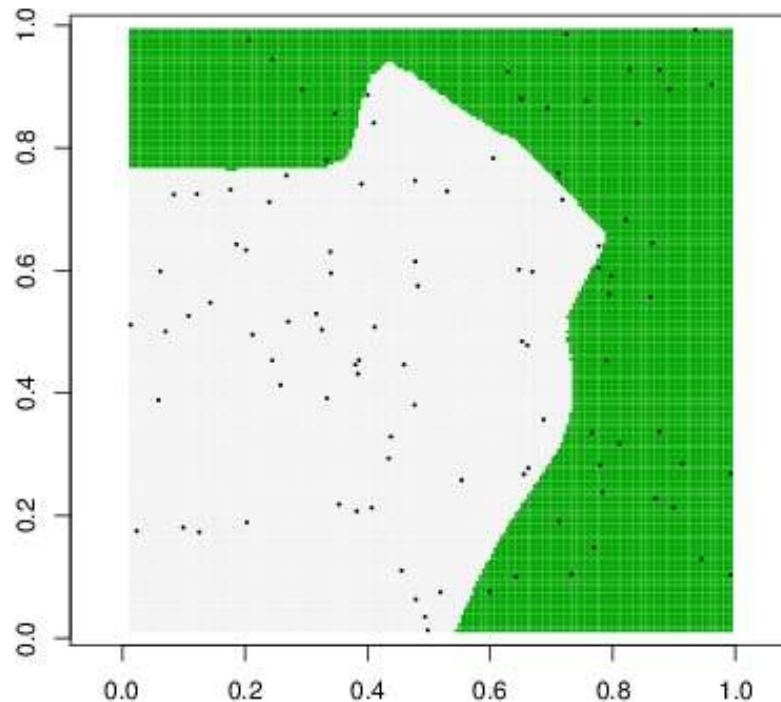
Posterior mode of population membership

4.5.3   F statistics

F statistics Weir and Cockerham [1984] relative to estimated clusters are obtained by:

```
Fstat.output(genotypes=geno,path.mcmc="./")
```

   which returns:

```
$Fis
[1] 0.1427044 0.1512689

$Fst
          [,1]       [,2]
[1,] 0.00000000 0.03562966
[2,] 0.03562966 0.00000000
```

   namely, individual F_{IS} and pairwise F_{ST} for estimated clusters.
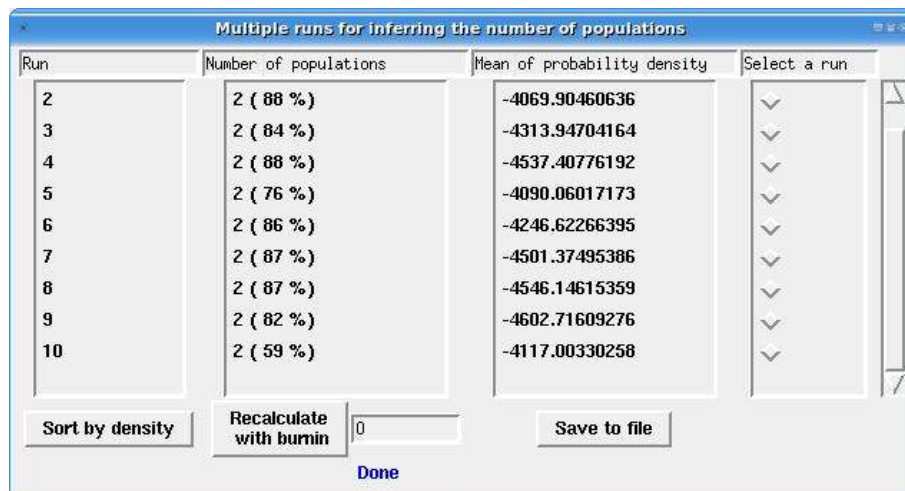
4.6 Launching several independent runs

Inference is based on a stochastic method, i.e. for a given dataset, the values of estimated parameters are random and depend on what happened during the run. In theory, different runs should give approximately the same estimates provided they are long enough. A good way to check that a run was long enough is to launch different runs and check that they provide approximately the same parameter estimates (K, individual population membership, maps).

This can be done automatically by clicking on the `Multiple independent runs` options:
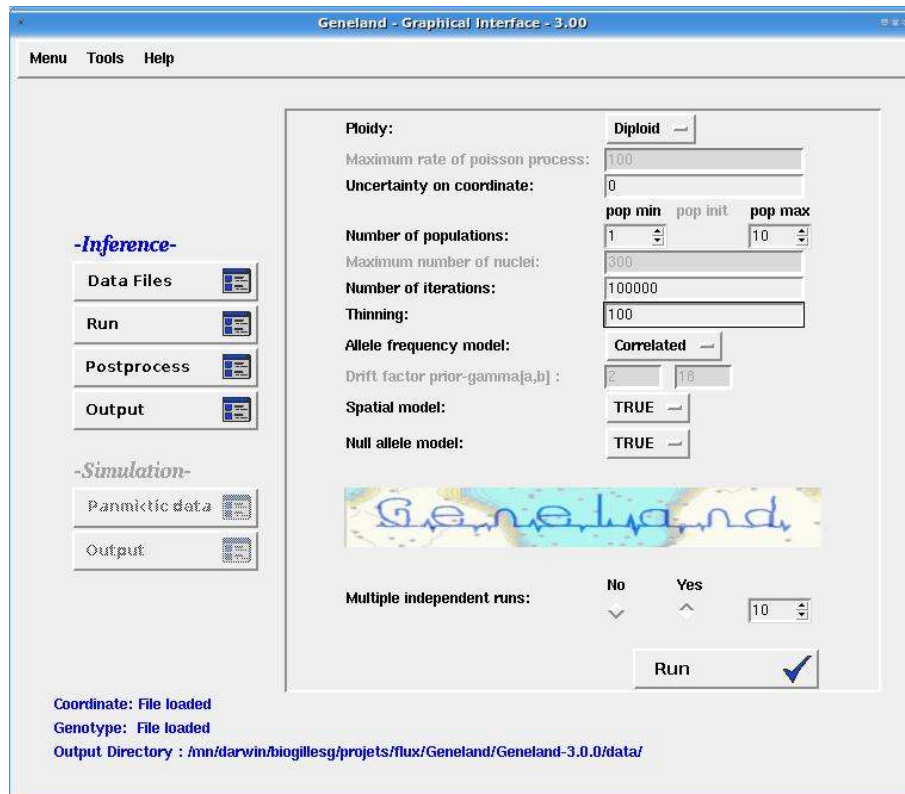
Runs are launched sequentially and computations are monitored in a new window:

Here, the results of computations are consistent across runs in terms of estimated number of populations K. If different runs give different results, it is recommended to base conclusion on the run giving the highest average posterior probability (run # 2 above).

The MCMC output of these run is saved in the output directory. The output of each run is saved in a separate directory (named 1,2,...,). The global ranking of the runs can also be saved in a file.

4.7 Parallel computations

It is possible to take advantage of clusters and multi-core computers when running several independent runs. The supported parallelisation systems are MPI (Message Passing Interface) and PVM (Parallel Virtual Machine).

In order to use this feature you need to install the R packages `snow` and `Rmpi` or `rpvm`. This can be done for example via the R command line by typing: `install.packages("snow")` and `install.packages("Rmpi")` or `install.packages("rpvm")`.

Before starting the multiple runs go to the menu Tools and select the parallel processing. You should see a window like this:
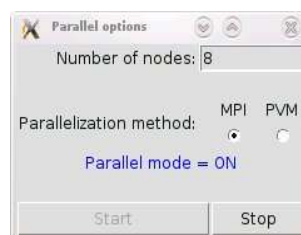
Here you choose the number of nodes (> 1) and the parallelisation system (MPI or PVM). Then click start, when the message sets to ON then you can start your multiple run. Note that since inference is based on MCMC a single run can not be run on several processors.

5   Example of data analysis using the R command-line

5.1   Preliminary steps

5.1.1   Organising your session

If you plan to work through the R command-line, do not type directly the command in the prompt. Type them first in a data editor (the built-in command line editor under Windows or emacs under Linux). This allows you to keep a trace of your work not only as numerical output but also as something looking like a computer program that you can re-use, correct, modify, share later. In addition, storing R code often takes far less disk space than storing the numerical output.

5.1.2   Launching GENELAND

Assuming R and GENELAND are installed,

- launch R

- you can launch the on-line help by typing `help.start()` in the R prompt (optional but very useful)

- load GENELAND by typing `library(Geneland)` in the R prompt

- under Mac-OS, make sure that X11 is launched (see section 1.3.2).

5.1.3   Loading the data

Let us assume that the data file(s) are named `genotypes.txt` and `coordinates.txt` and stored in a directory called `data`. Then type in the R prompt:

```
geno <- read.table("../data/genotypes.txt",na.string="000") ## change "000" to according to your fil
```

... the genotypes are loaded and stored in an R object called `geno`.

Then type again in the R prompt:

```
coord <- read.table("../data/coordinates.txt")
```

... the coordinates are loaded and stored in an R object called `coord`.

Generally you can replace `../data` by any string `path_to_my_data` giving the path to the data relatively to the working directory. Under Windows this working directory is specified through the menu file and sub-menu preferences. Under Linux, the R working directory is the Linux working directory of the terminal from which R was launched.

5.1.4   Checking the data

You can control that you have correctly loaded your data by typing the names in the prompt, e.g.:

```
> coord[1:10,]
```

will print the ten first lines of the coordinates.

The objects are a bit too large to be visualised in the R shell, it is more convenient to watch them through the built-in data editor:

```
fix(coord)
```

or

```
fix(geno)
```

you can check the dimension of the object:

32

```
dim(coord)
```

...we have indeed one line per individual and two columns, and...

```
nrow(geno)
```

...one line per individual and ...

we have indeed one line per individual and two columns, and...

```
ncol(geno)
```

two columns per locus for dipoloid data.

You can also plot the coordinates by

```
plot(coord,xlab="Eastings",ylab="Northings",asp=1)
```

...which opens a new window with the desired plot.

5.2 Inference

We now carry out an analysis to infer the number of populations and their spatial boundaries for this dataset. This is done with the main GENELAND function named MCMC. A possible call of this function is as follows:

```
MCMC(coordinates=coord,
     geno.dip.codom=geno,
     varnpop=TRUE,
     npopmax=10,
     spatial=TRUE,
     freq.model="Correlated",
     nit=100000,
     thinning=100,
     path.mcmc="./")
```

This will perform parameter inference by MCMC simulation assuming that

- the number of HWLE populations is unknown and hence treated as simulated variable along the MCMC simulations (`varnpop=TRUE`)

- but smaller than 10 (`npopmax=10`),

- using the spatial model (`spatial=TRUE`),

- combined with the correlated frequency model (`freq.model="Correlated"`).

- the number of MCMC iterations will be 100000 (`nit=100000`)

- and only each 100th iteration will be saved on the disk (in total, 1000 iterations will be saved)

- in the current working directory (`path.mcmc="./"`).

This function takes many more arguments, most of them being optionals (i.e. with default values). An on-line help for function MCMC is given by typing ? MCMC. See section 7.

5.3   Post-processing MCMC outputs

The call to function MCMC generates different files in the directory specified by the argument `path.mcmc`. Information is extracted from these files through a call to function `Post.Process.Chain` as follows:

```
PostProcessChain(coordinates=coord,
                 geno.dip.codom=geno,
                 path.mcmc="./",
                 nxdom=100,
                 nydom=100,
                 burnin=200)
```

This will create additional files required to get final estimates and maps. In the example above, we have:

- an horizontal discretization of the study domain in 100 pixels (`nxdom=100`)

- a vertical discretization of the study domain in 100 pixels (`nydom=100`)

- a burn-in of 200 saved iterations, i.e. discarding the 200 first saved iterations (`burnin=200`)

5.4   Generating graphical and numerical outputs

5.4.1   Estimated number of HWLE populations

The number of population simulated from the posterior distribution can be visualised by:

```
Plotnpop(path.mcmc="./",
         burnin=200)
```

This produces a plot like
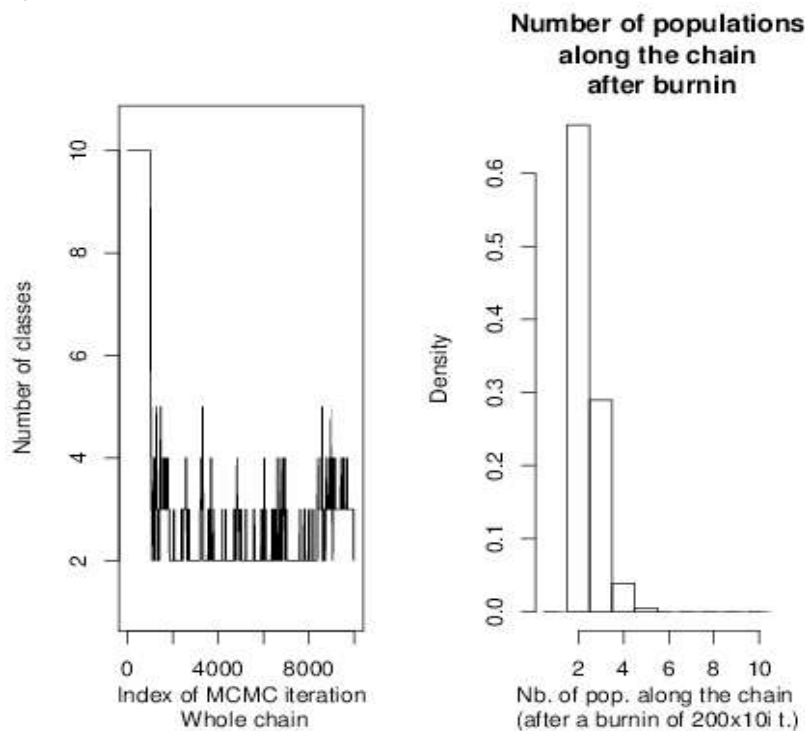
this run displays a clear mode at $K = 2$ and a relatively good mixing around this value.

5.4.2 Saving graphics

GENELAND functions for graphics contain arguments that allows users to send save graphics as pdf or postscript files. Here is an example:

```
Plotnpop(path.mcmc="./,
burnin=200,printit=TRUE,
file="Number_of_Clusters.pdf",format="pdf")
```

5.4.3   Map of posterior probability of population membership

A call to function `PosterioMode` like:

```
PosteriorMode(coordinates=coord,
              path.mcmc="./",
              file="map.pdf")
```

will produce a plot like:

Posterior mode of population membership
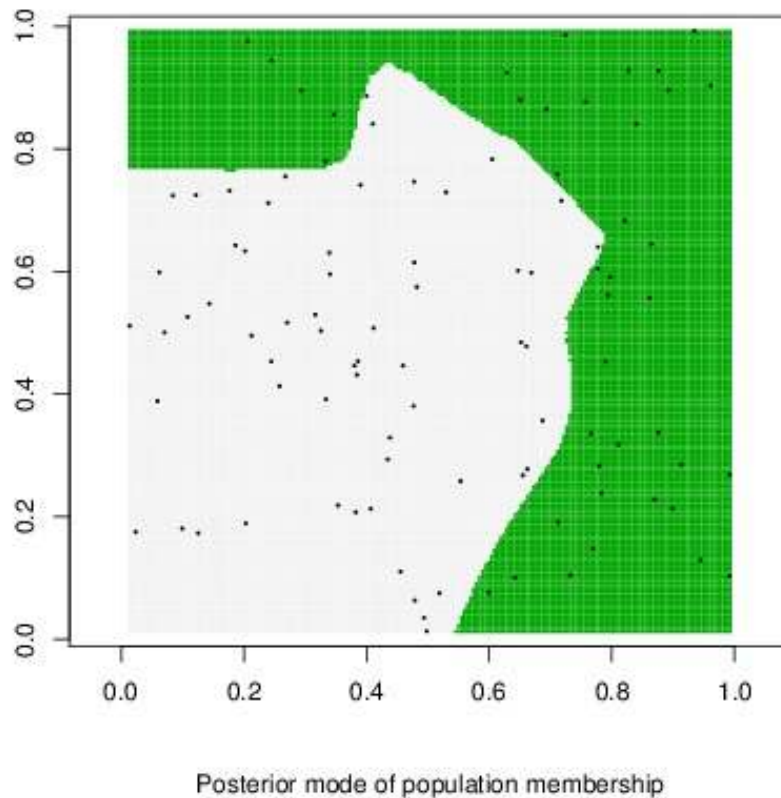
5.4.4   F statistics

F statistics relative to estimated clusters are obtained through

```
Fstat.output(genotypes=geno,path.mcmc="./")
```

which returns:

```
$Fis
[1] 0.1427044 0.1512689

$Fst
           [,1]       [,2]
[1,] 0.00000000 0.03562966
[2,] 0.03562966 0.00000000
```

namely, individual F_{IS} and pairwise F_{ST} for estimated clusters.

5.5 MCMC convergence assessment

5.5.1 Checking MCMC convergence, what does that mean?

When it comes to Markov chains, convergence means that after enough iterations, the simulated vector is sampled from the desired target distribution. Since this target distribution is highly multi-dimensional and not much is known about it, checking convergence is not something easy to do and is actually most often impossible. The best that can be done is to check that there are no obvious clues indicating a lack of convergence. MCMC behaviours indicating a lack of convergence include:

- a single chain displaying a transient behaviour, for example a clear decreasing or increasing trend from its initial value

- multiple chains leading to different estimations (number of population K, population memberships)

- a single chain stuck at a particular value or in a particular interval. Note however that in the case of a single integer parameters (for instance, the number of populations), a chain stuck at a particular value can be either a genuine feature of the posterior distribution or convergence flaw.

See also an interesting page by Charlie Geyer advocating the use of a few long runs against many short runs: `http://www.stat.umn.edu/~charlie/mcmc/one.html`.

5.5.2 Factors affecting MCMC convergence

We have observed that mixing properties and hence convergence are affected by

- the number of individuals n, with poorer mixing properties as n increases

- the number of loci L, with poorer mixing properties as L increases

- departure from model assumptions. In case the dataset does not consist of genuine HWLE groups, it seems that the posterior distributions often exhibits complex patterns of local modes in which MCMC simulations get more easily trapped.

5.5.3 Example of R code as an aid to convergence diagnostic

As explained above about the multiple runs option in the GUI, MCMC convergence should be checked by comparing the output of several independent runs.

This can be done manually through a loop as follows:

```
## Loop for multiple runs
nrun <- 10
burnin <- 200
for(irun in 1:nrun)
  {
    ## define path to MCMC directory
    path.mcmc <- paste("./",irun,"/",sep="")
    system(paste("mkdir ",path.mcmc))
    MCMC(coordinates=coord,
         geno.dip.codom=geno,
         varnpop=TRUE,
         npopmax=10,
         spatial=TRUE,
         freq.model="Correlated",
         nit=100000,
         thinning=100,
         path.mcmc=path.mcmc)
```

```
    ## MCMC postprocessing
    PostProcessChain(coordinates=coord,
                     path.mcmc=path.mcmc,
                     nxdom=200,
                     nydom=200,
                     burnin=burnin)
  }



## Computing average posterior probability
## with a burnin of 200 (* 100) iterations
lpd <- rep(NA,nrun)
for(irun in 1:nrun)
  {
    path.mcmc <- paste("./",irun,"/",sep="")
    path.lpd <- paste(path.mcmc,"log.posterior.density.txt",sep="")
    lpd[irun] <- mean(scan(path.lpd)[-(1:burnin)])
  }



## Runs sorted by decreasing average posterior probability:
order(lpd,decreasing=TRUE)
```

6   Assessing influence of modelling assumptions

6.1   Choosing a model to perform MCMC simulations

Computations are obtained under specific assumptions regarding allele frequencies (correlated/ non correlated) and population membership (spatial / non spatial). First you have to chose which combination of models is the most suitable for your data. Roughly, if you expect differentiation due to the presence of simple shaped landscape features, the spatial model is presumably well suited. And if you are looking for low differentiation due to recent ecological events, the correlated allele frequencies model is more suitable.

6.2   Comparing outputs from MCMC runs under different models

The estimates under your preferred model should be compared to estimates under other models. Note that even if a model is more realistic than others from the biological point of view, the results of analysis under this "best" model can be tricked by poor MCMC mixing. This can be observed on large datasets (> 1000 individuals, $> 100 loci$) and/or due to departure from modelling assumptions.

The most comfortable situation is when different models give similar answer. In this case, there is presumably a strong signal in the data and the inferred pattern does not depend on the particular way information is extracted (model+algorithm).

In case results differ across models, our recommendations are as follows:

- check convergence under the different models

- give preference to models that fits better with the organism under study

  - *a priori*: in the sense of prior knowledge about dispersal, potential barriers to gene flow...

  - *a posteriori*: in the sense where estimated K and maps complies best with what is known about the organism

- do not attempt to compare different models on the basis of the average posterior probability. Indeed, they are defined on different parameter spaces and such a comparison do not make sense mathematically.

6.3   Objective criterions to perform model selection [TODO]

7   More examples using the R command-line

7.1   Estimating frequency of null alleles

If the option for filtering null alleles is chosen for simulation with function `MCMC`, the estimated frequency of null alleles at each locus can be obtained through function `EstimateFreqNA` as follows:

`EstimateFreqNA(genotypes=geno,path.mcmc="./")`

which returns a vector of length `nloc` (the number of loci) whose entries are estimated frequencies of null alleles.

7.2   Analysing Geo-referenced data with a non-spatial prior

A dataset consisting of Geo-referenced genetic data can be analysed with a non-spatial prior for population membership. The coordinates are not used in the inference algorithm, they are just used for the graphical representations.

Similarly to the example of R code given in section 5.2, this can be done by:

```
MCMC(coordinates=coord,
    geno.dip.codom=geno,
    varnpop=TRUE,
    npopmax=10,
    spatial=FALSE,          ## the argument spatial is now set to FALSE
    freq.model="Correlated"
    nit=100000,
    thinning=100,
    path.mcmc="./")
```

7.3   Analysing non spatial data

It is also possible to analyse datasets consisting only of genotypes (no spatial coordinates). This can be done by a call of function MCMC as:

```
MCMC(geno.dip.codom=geno,
    varnpop=TRUE,
    npopmax=10,
    spatial=FALSE,          ## the argument spatial set to FALSE
    freq.model="Correlated"
    nit=100000,
    thinning=100,
    path.mcmc="./")
```

This allows to estimate the number HWLE populations and population memberships of individuals. Spatial graphical displays do not make sense in this context.

7.4   Getting improved graphics

You may find that the graphical features available through the GUI and the GENELAND graphical functions are too limited. The possibilities to improve graphics by working directly though the command-line are almost unlimited.

7.4.1   Superimposing countries boundaries and coast lines on a GENELAND **map**

Let us assume that you have a map giving population memberships or posterior probabilities of population memberships over a given domain. You can superimpose countries boundaries and coast line by typing:

`map(resolution=0,add=TRUE)`

This will add the desired lines to the active graphic window (by default the last window opened). The active graphic window can be redefined by e.g. `dev.set(3)` (this will set window # 3 as active window). See on-line help of function map for details (`? map`).

7.5   Exploring MCMC outputs to better check convergence [TODO]

7.6   R and Geographical Information System (GIS) data

There exits many sites and specialised packages that you may find useful to handle your GIS data and combine them with GENELAND outputs. In this ection, we provide some pointers towards useful pages and try to a give a flavour of what is possible under R.

7.6.1   Resources

On the various formats and how to load them in R: `http://help.nceas.ucsb.edu/R:_Spatial`

Loading data, converting coordinates, plotting: `http://sites.google.com/site/spatialr`

Data (climate, landcover, coastal lines, administrative boundaries: `http://www.fao.org/geonetwork`

7.6.2   Loading and plotting coastal lines

An example with the FAO World Food Summit Map Coastline and International boundary (dataset available as a shapefile named wfs_coas.shp from `http://www.fao.org/geonetwork`).

```
library(maptools)
map <- readShapePoly("wfs_coas.shp")
plot(map,
     ## uncomment optionnal window limits
     # xlim=c(-5,15),ylim=c(40,60)
     )

library(maptools)
map <- readShapePoly("wfs_coas.shp")
plot(map,
    ## uncomment optionnal window limits
    # xlim=c(-5,15),ylim=c(40,60)
    )
```

7.6.3   GRASS [TODO]

Output of GENELAND can be combined with high quality maps obtained with the Geographic Resources Analysis Support System, commonly referred to as GRASS. See `http://grass.osgeo.org` and `http://grass.osgeo.org/wiki/Main_Page`. Some of the tasks can be performed directly under R through the R package spgrass6. See `cran.r-project.org/web/packages/spgrass6`.

7.6.4   Superimposing R plots on Google maps

The R package RgoogleMaps makes it possible to download Google maps, display and save them under R and superimpose other R grahical objects. This requires R package gdal and libraries gdalconfig and proj[4]
Here is a small example. Assume you have a matrix of spatial coordinate named coord looking like:

```
> coord
X        Y
13.67137 55.60033
13.62235 56.47770
13.49812 55.95658
17.44374 60.33607
17.44306 60.32254
16.08196 59.88587
16.08384 60.11025
20.98510 68.10043
20.88659 67.65393
22.17869 68.05858
```

---

[4]To be installed nder Ubuntu with sudo apt-get install gdalconfig proj. Not tested under Windows.

```
        21.47275 67.63452
        20.43166 68.34153
        23.08695 67.11904
        22.51499 67.14419
        17.83787 63.15781
        17.76166 63.27629
```

Dowload the map:

```
MyMap.zoom8 <- GetMap(center=apply(coord[,2:1],2,mean), # center of area
                                                        # (needs Lat/Lon
                                                        # not Lon/Lat!)
                      zoom =8,  # resolution
                      maptype = "satellite")
```

To get a plot as in figure 8, type:

```
PlotOnStaticMap(MyMap.zoom8,
                lat=coord[,2] , lon= coord[,1] ,col='red')
```

Much more on the on-line help of R package RgoogleMaps (? RgoogleMaps).

Figure 8: A rare sunny view of the Oslo fjord at two resolutions!

8   Simulation of data under the spatially organised HWLE populations model

Datasets can be simulated under the statistical models described in section 2. This can be done via the R shell by calling function simFmodel as follows:

```
simdata <- simFmodel(nindiv=100,
                     coord.lim=c(0,1,0,1),    ## simulations on the unit square
                     number.nuclei=15,        ## tessellation driven by 15 polygons
                     nall=rep(10,20),         ## 20 loci with 10 alleles each
                     npop=3,                  ## 3 populations
                     freq.model="Correlated", ## Correlated frequency model
                     drift=rep(0.04,3),       ## drift (or Fst) parameters
                     dominance="Codominant"   ## codominant-like genotypes
                     )                        ## (two columns per locus)
```

Some of the arguments are optional. See on-line help (? simFmodel) for details.

The R object simdata is a list whose components can be checked by summary(simdata). The genotypes are stored as simdata$genotypes and the coordinates as simdata$coordinates and are suitable for simulations studies, e.g. to assess the effect of number of loci, number of individuals, allele diversity … on accuracy of inferences.

9   Simulation of data with spatially auto-correlated allele frequencies

There is a function for simulation of data under the model described in [Guillot and Santos, 2009]. This function is called `simdata` and is an extension of function `simFmodel` described in the previous section. It allows to simulate genotypes for individuals that are structured by isolation-by-distance and barriers to gene flow. The function takes many arguments, most of them being optionnal.

See on-line help (? `simdata`) for detail about these arguments.

Here is an example for simulation of genotypes of 100 individuals at 3 loci with 5 alleles at each locus (values artificially small to limit the number of graphics generated in the next step).

```
dataset <- simdata(nindiv=100,
                number.nuclei=10,
                allele.numbers=rep(5,3),
                model="stable",
                IBD=TRUE,
                alpha=1,
                beta=1,
                gamma=1,
                npop=3,
                give.tess.grid=TRUE,
                give.freq.grid=TRUE,
                npix=c(100,100),
                comp.Fst=TRUE,
                comp.Dsigma2=TRUE,
                comp.diff=TRUE,
                width=0.1,
                plot.pairs.borders=FALSE)
```

The function returns a list stored in the R object `dataset` . Information about the object can be obtained by the function `summary` as:

```
summary(dataset)
```

The coordinates of individuals can be referred to by

```
dataset$coord.indiv
```

The genotypes of individuals can be referred to by

```
dataset$genotypes
```

The cluster membership of the individuals can be referred to by

```
dataset$color.nuclei[dataset$nearest.nucleus.indiv]
```

The simulated dataset can be visualised by function `show.simdata` e.g. by

```
 show.simdata(dataset,
     plot.coord = TRUE,
     plot.tess = TRUE,
     plot.freq.grid = TRUE,
     loc.grid = 1,
     zlim.freq=c(0,1))
```

This will produce a plot of the coordinates of individuals, a map of the tessellation induced by the barriers simulated and maps of allele frequencies for alleles at the first locus (`loc.grid=1`). See on-line help (? `show.simdata`) for details.

10   Using other softwares to analyse GENELAND outputs

10.1   Population genetics softwares

10.1.1   Genepop

There is a GENELAND function called gl2gp that writes coordinates and genotypes into an ascii file suitable for analysis with the GENEPOP program [Rousset, 2007]. See on-line help (? gl2gp) for details. The genotype file produced might require some extra hand editing.

10.2   MCMC post-processing softwares

10.2.1   Partitionview

TODO

10.2.2   Distruct

Distruct can be used to visualise the distribution of individual population membership. The information required as input for this program is in the file proba.pop.membership.indiv.txt stored in the GENELAND output directory. See section C.2.3. The information displayed is somehow similar to what is obtained with the GENELAND function PlotTessellation though disregarding the spatial aspect of the dataset.

11   Analyzing other software's outputs [TODO]

A   Frequently asked questions

A.1   Can you provide sample input files?

There are to be found in the directory `data` of the Geneland distribution archive file.

A.2   Can I use population data?

It is possible to analyse datasets where different individuals share the same spatial location. If such data are treated without *uncertainty on coordinate* in the GUI or `delta.coord` is set to 0 in function `MCMC` then individuals sharing the same coordinates will be assigned to the same inferred group.

A.3   Can I use SNPs?

It is possible to run GENELAND with SNPs. The bases have to be recoded as $\{1, 2, 3, 4\}$. Fixed alleles are not allowed. Loci carrying such alleles have to be removed from the dataset first.

A.4   How to deal with datasets containing a large number of loci?

A.4.1   Running GENELAND on a random subsets of loci

You can run GENELAND for various random subsets of loci with e.g. 500-1000 loci and check that the different subsets give you the same answer. If this happens, you can confidently infer that runs with a larger number of SNPs would also give you the same answer. If different subsets give different answers, it can be due either to convergence issues (the genuine pattern contained in the various subsets is the same but GENELAND failed to detect it because of numerical difficulties) or to an excess of sampling variance in the selection of loci at random. There is no obvious way to decipher the relative importance of those two factors.

A.4.2   Managing output files

By default, the Markov chain of all parameters are saved on the disk. The biggest files are usually those containing estimated allele frequencies (`frequencies.txt` and `ancestral.frequencies.txt`) and coordinates of nuclei of the Voronoi Tessellation (`coord.nuclei.txt`). It is recommended to compress or erase them after relevant information has been extracted from the corresponding runs. This requires to write a bit of R code (see e.g. `?  file.remove` or `?  file`).

Multiples runs via the GUI should be avoided as it will quickly generate huge files.

A.5   Can I use haploid data?

Yes. Specific computing options are implemented in GENELAND for haploid data.

A.6   Can I study organisms with ploidy other than haploid or diploid?

No. The only ploidy handled as of today are haploid and diploid.

A.7   Can I use dominant markers?

Yes.

A.8   Is there any way to account for the presence of special landscape features?

We have been often asked how to include information about the presence of "an urban area not suitable for my organisms" or "a land mass in the middle of sea water obviously not suitable for fish". There is currently no obvious way to do that with GENELAND. We recommend to analyse your data ignoring the presence of such landscape feature and to try to take it into account at the post-processing stage.

A.9   Can I study an organism leaving in a linear habitat?

Yes. The neat way to do that in GENELAND is to convert 2-dimensional coordinates into 1-dimensional curvilinear coordinates measuring distance of each individuals to an arbitrary origin in this habitat.

A.10   MCMC, burn-in, thinning... What does that mean?

MCMC stands for Markov chain Monte Carlo. It is a technique to simulate random things in a probabilistic model. It appeared in statistical physics in the 50's and in statistics in the 70's and was popularised at the end of the 80's in particular to make inference in complex Bayesian models. The key idea is that you have made assumptions about data and parameters summarised as probility distributions π(parameters), π(data | parameters). These two probability distributions can be combined to get the posterior distribution π(parameters | data) via Baye's theorem. See Beaumont and Balding [2004] for a recent review.

Although the previous distribution is known, it is most often impossible to compute directly the parameter value that makes it maximal (the so-called maximum *a posteriori* estimate of this parameter). The reason is that this parameter lies in a very high-dimensionnal space. For example in GENELAND, the vector of parameters include allele frequencies, hence hundreds of unknown numerical values. The strategy to deal with such high-dimensionnal models consists in drawing at random values of the parameter from the posterior distribution and then extract the desired information from this sample.

Again, this task is not straighforward and the approach taken is an approximation. It consists in starting from an arbitrary value that is iteratively modified in such a way that after many iterations, the distribution of the simuated parameter is close to the posterior distribution. One has to run this iterative scheme long enough to be sure that the simulated distribution is close enough to the posterior distribution (and in particular that the result is not affected by the choice of this initial value). One usually discards the first iterations that correspond to the so-called *burn-in* period. The simulation process is iterative and a value at iteration $t + 1$ is obtained by modifying a value at time t. This results usually in a high auto-correlation of the simulated values. Consecutive simulated values are usually redundant. To save time and disk space, one usually *thin* the chain, i.e save only a fraction of simulated values.

A.11   How should I choose K_{max}?

Take it a bit larger that the largest value that you can reasonably expect for your dataset. You can diagnose that the value set for K_{max} was too small if the chain simulating the values of K (as displayed by function `Plotnpop`) get stuck at this maximum value. In such a case, it is recomended to re-run MCMC computations with a larger value for K_{max}. Conversely, a value K_{max} taken much larger than the true K (if any) does not bring any problem but it can slow down un-necessarily computations.

A.12   Which value should I choose for the number of MCMC iterations?

There is no obvious answer to that. The value should be large enough to avoid any of the symptoms of lack of convergence described in section 5.5.2. A rough order of magnitude would be $n_{it} = 100000$ for a dataset of $n = 100 - 300$ individuals at $L = 10 - 30$ loci.

A.13   Which value should I choose for the thinning?

The thinning is defined as the proportion of MCMC iterations saved on the disk. This computing option has a limited effect on the accuracy of inferences. It is more a matter of how many disk space is available. For a run of $n_{it} = 100000$ iterations, we typically save 1000 iterations and set the thinning to 100.

A.14   I have launched 50 runs of 5000000 iterations with a thinning of 1 and my disk is full!!

You don't need to save each single iteration. MCMC produce correlated samples. Saving one iteration out of 100 (`thinning=100`) is usually enough. The thinning can be increased even more as long as the number of iterations saved (`nit/thinning`) remains large ($100 - 1000$). Remember that the amount of disk space required to store results of MCMC iterations increases approximately linearly with the number iterations.

A.15   How does GENELAND treat a double missing genotypes under the "filter null alleles" scheme?

By default as a genuine null allele. See section 2.4.1 on how a distinction between null alleles and missing data can be made via argument `miss.loc` of function `MCMC`.

A.16   What is the difference between running the model with coordinates under the non-spatial model and without coordinates at all, and how can I implement these computing options?

If you have continuous spatial sampling (one individual only per site) then the `spatial=FALSE` option will give the same result with and without coordinates. In that case, you can use directly the GUI with coordinates. If you have "clumpped" spatial sampling (or population data, i.e. several individuals sharing the same coordinates) inputing the coordinates under the `spatial=FALSE` will return clusterings where all individuals at the same site belong to the same cluster. This is equivalent to the USEPOPINFO option of the STRUCTURE program. If you want to disregard completely any spatial information, then you can skip the `coordinates` argument in function `MCMC` and set `spatial =FALSE`. To implement the same computing option in the GUI, you have to input a file of dummy coordinates with no individuals sharing common coordinates , e.g. as

```
nindiv <- nrow(genotypes)
n.int <- ceiling(sqrt(nindiv))
x <- rep(seq(from = 0, to = 1, length = n.int), n.int)
y <- rep(seq(from = 0, to = 1, length = n.int), n.int)
y <- as.vector(t(matrix(nr = n.int, nc = n.int, y,byrow = FALSE)))
coordinates <- cbind(x, y)[1:nindiv, ]
```

write this R object as a text file and use t under the non-spatial option of the GUI.

A.17   What about the two-step procedure described by Guillot et al. [2005a]?

Guillot et al. [2005a] recommended to perform inferences in two steps: a first run with K treated as unknown and variable to estimate it and a second run with K fixed at the value estimated in the first run to estimate individual cluster memberships.

   Since version 2.0, the algorithm has been modifed and inferences should be done in a single step [Guillot, 2008]. This makes inferences simpler, faster and it avoids the issue of *"ghost populations"* reported by Guillot et al. [2005a]. One should still keep in mind potential MCMC convergence issues and check that parameters inferred are consistent across several such single-step runs.

A.18   How to select a run among several runs performed under the same model options?

To compare outputs of several runs under the same model options, use the mean posterior density. Note that the posterior density of the inferred K (given as a percentage in the output under the GUI) should not be used as a measure of goodness of fit.

A.19   How to select a run among several runs performed under different model options?

Comparing outputs of different models is a difficult issue in statistics and there is no satisfactory solution for clustering models in populations genetics as of today. The best you can do is to check that the pattern(s) you get under the different models pop up consistently over several runs. If this holds, check that the inferred clusters comply with model assumptions. In GENELAND, this amounts to check that inferred clusters do not depart significantly from HWLE and that inferred clusters are significantly differentiated (e.g. with GENEPOP). This step can be time-consuming. If an inferred pattern passes this check, there is good chance that this pattern is real (and not a model/algorithm artefact). If several patterns passe this check, they should look alike. If not, this might be indicative of the violation of some of the modelling assumptions. See also Guillot et al. [2009] for further discussion.

A.20   I only have one set of coordinates per sampled population, not individual. Can I set some uncertainty on the coordinates?

Without uncertainty on the coordinates, all individuals sampled at the same site will be assigned to the same cluster. Setting a non zero value for the uncertainty on coordinates has the feature (among others) to allow GENELAND to assign individuals sampled at the same site to different clusters. In case of good compliance of the data to the model (well differentiated HWLE clusters), this feature might help to detect a few migrants that might remain undetected otherwise. See also section 2.4.2.

A.21   Any recommendations as to the parameter to use for uncertainty on the coordinates?

The idea behind the uncertain coordinates is that the observed coordinates are equal to some true coordinates plus a random error following a uniform distribution. The value of the parameter passed to GENELAND is the length of edge of the square of this uniform distribution. The most relevant value for this parameter depends on the actual uncertainty on coordinates and on the species and the size of the sampling area. There is no absolute recommendation as to the value of this paremeter. See also section 2.4.2.

A.22   Power, required sample size, etc...

One may wonder what is the smallest genetic differentiation (measured e.g. in terms of F_{ST}) detectable by GENELAND. The answer depends on the number of individuals in the sample, the number of loci, the level of polymorphism of the markers used, the number of individuals of each source populations present in the sample, on the proportion of errors in the data (e.g. null alleles) whether markers are co-dominant or dominant and last but not least, on how well the model assumed by GENELAND fits the data. In best case situations, we were able to detect the presence of two populations displaying an F_{ST} of the order of 0.01 from a sample of 100 individuals with 20 loci. See supplementary material in Guillot et al. [2008] for details and futher numerical results.

   Note that detecting a unkwnon structure is a much more difficult statistical issue than just testing whether two populations are differentiated or not. See example in figure 9 for an illustration with a quantitative variable measured in two populations.

A.23   "Ghost" clusters

GENELAND outputs include an estimate $\hat{K}$ of the number of clusters present in the sample and a map of the geographical locations of these clusters. It might happen that $\hat{K}$ is larger than the number of clusters observed on this map. The non-observed clusters where referred to as "ghost" clusters by Guillot et al. [2005a]. The first version of GENELAND Guillot et al. [2005a], Guillot [2005] was prone to this issue which was mostly due to a lack of MCMC mixing corrected in Guillot et al. [2008]. This phenomenon can still occur in some instances. Imagine for example a set of three individuals for which inference is carried out with an MCMC run of three iterations. Denoting $p^t_{i=1,2,3}$ the simulated cluster membership of individual i at iteration t and assuming that the chain simulated is $p^1 = (2,1,1)$, $p^2 = (1,2,1)$ and $p^3 = (1,1,2)$. The number of clusters simulated along the chain is equal to 2 along the whole run. So will be the estimated K. However, the three individuals will be assigned to cluster 1. For data complying well with the modelling assumtptions of GENELAND, and after relabelling the simulated cluster memberships[5], such inconsistency on the estimated K was extremely rare. Occurence of this issue might be an indication of departure of the data from modelling assumptions.

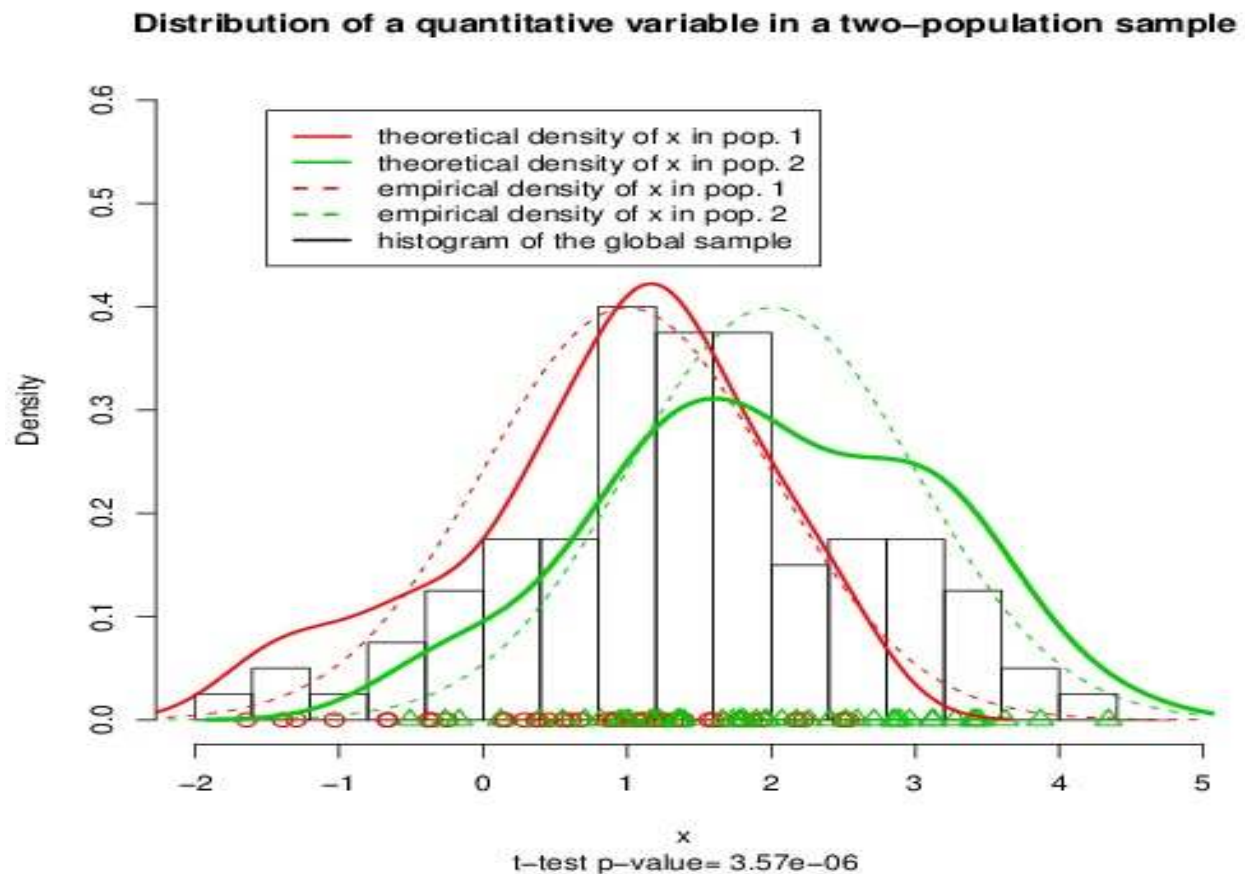[5]as explained in Guillot et al. [2008] and performed in GENELAND since version 3.0.0

Figure 9: Synthetic example of a variable measured in two popualtions with mean equal to 1 and 2 respectively and a common unit variance. The global histogram (black) seems unimodal and does not suggest the presence of two populations, the numerical values from the two population overlap substantially. Any algorithm aimed at clustering the individuals would be subject to a large error rate. However, the p-value of a classical t-test is highly significant.

A.24   Non recombining DNA sequences

DNA sequence data can be analyzed with GENELAND provided they are stored in the proper format (the bases have to be recoded as $\{1, 2, 3, 4\}$, see section 3). Computerwise, analyzing *non recombining* DNA sequences does not bring up any difficulty. From a statistical/genetic point of view, the meaning and potential outcome of this kind of analyis is less straighforward. GENELAND assumes linkage equilibirum, an assumption violated in absence of recombination. One may expect in some instances that the inferred clusters mirror the spatial location of the main haplogroups. This has not been assessed theoretically or empirically as of today.

B   Algorithm

B.1   Simulation based inference [TODO]

B.1.1   Special aspects

K_{init} set to K_{max} during the first iterations

B.2   Post-processing MCMC outputs [TODO]

B.2.1   Estimating K

B.2.2   Dealing with label switching

B.2.3   Computing posterior probability of cluster memberships

B.3   F statistics [TODO]

- Estimation according to Weir and Cockerham [1984]
- Missing data allowed. Induce a small bias in the estimation of F_{ST} and a somehow larger bias in the estimation of F_{IS} .
- Not implemented for haploid data

C   Description of MCMC output files

C.1   Files produced by MCMC simulations

Note that some of those files are not created by default. See on-line help for detail.

C.1.1   `parameters.txt`

List of characteristics of the dataset and all arguments passed to function *MCMC* (via the GUI or directly).

C.1.2   `populations.numbers.txt`

Simulated values of number of populations K.

C.1.3   `nuclei.numbers.txt`

Simulated values of number of Voronoi cell m coding for cluster membership. (Set to n under the non-spatial option).

C.1.4   `coord.nuclei.txt`

Simulated coordinates of the nuclei of Voronoi cells in the tessellation uses to parameterise cluster membership. (Set to the coordinates of sampled individuals under the non-spatial option).

C.1.5   `color.nuclei.txt`

Simulated cluster membership of Voronoi cells. It is coded as an integer and displayed as a color.

C.1.6   `ancestral.frequencies.txt`

Simulated allele frequencies of ancestral cluster in the correlated allele frequencies model.

C.1.7   `drifts.txt`

Simulated drift parameter (or F_{ST}) in the correlated allele frequencies model.

C.1.8   `frequencies.txt`

Simulated allele frequencies of present time clusters in the correlated allele frequencies model.

C.1.9   `hidden.coord.txt`

Estimated "true coordinates" if some uncertainty on coordinates is assumed.

C.1.10   `log.likelihood.txt`

Log-likelihood along MCMC simulation.

C.1.11   `log.posterior.density.txt`

Log of posterior density of simulated parameters along MCMC simulation.

C.2   Files produced when post-processing MCMC simulations

C.2.1   `postprocess.parameters.txt`

List of all arguments passed to function *PostProcessChain* (via the GUI or directly).

C.2.2   `proba.pop.membership.txt`

Posterior probability of cluster membership for pixels of a discretization of the domain.

C.2.3   `proba.pop.membership.indiv.txt`

Posterior probability of cluster membership for sampled individuals.

C.2.4   `modal.pop.txt`

Estimated cluster membership for pixels of a discretization of the domain.

C.2.5   `modal.pop.indiv.txt`

Estimated cluster membership for sampled individuals.

C.2.6   `perm.txt`

Permutation of cluster labels allowing to get rid of the label switching issue.

References

D.J. Balding. Likelihood-based inference for genetic correlation coefficients. *Theoretical Population Biology*, 63:221–230, 2003.

M.A Beaumont and D.J. Balding. Identifying adaptive genetic divergence among populations from genome scans. *Molecular Ecology*, 13:969–980, 2004.

D. F. Callen, A.D. Thompson, Y. Shen, H. A. Phillips, R. I. Richards, and J. C. Mulley. Incidence and origin of 'null' alleles in the (ac)n microsatellite markers. *American Journal of Human Genetics*, 52:922–927, 1993.

A. Coulon, G. Guillot, J.F. Cosson, J.M.A. Angibault, S. Aulagnier, B. Cargnelutti, M. Galan, and A.J.M Hewison. Genetics structure is influenced by lansdcape features. Empirical evidence from a roe deer population. *Molecular Ecology*, 15:1669–1679, 2006.

E.E Dakin and J.C. Avise. Microsatellite null alleles in parentage analysis. *Heredity*, 93(5):504–509, 2004.

M. Foll, M.A. Beaumont, and O. Gaggiotti. An approximate Bayesian computation approach to overcome biases that arise when using AFLP markers to study population structure. *Genetics*, 179:927–939, 2008.

M. Fontaine, S.J.E. Baird, S. Piry, N. Ray, K. Tolley, S. Duke, A. Birkun, M. Ferreira, T. Jauniaux, A. Llavona, B. Öztürk, A.A. Öztürk, V. Ridoux, E. Rogan, M. Sequeira, U. Siebert, G.A. Vikingson, J.M. Bouquegneau, and J.R. Michaux. Rise of oceanographic barriers in continuous populations of a cetacean: the genetic structure of harbour porpoises in old world waters. *BMC Biology*, 5 (30), 2007.

B. Guedj and G. Guillot. Estimating the location and shape of hybrid zones. *To appear in Molecular Ecology Resources*, 2011. 16pp.

G. Guillot. Geneland: a computer package for landscape genetics. Contributed R package, 2005. http://www2.imm.dtu.dk/ gigu/Geneland/Geneland-Doc.pdf.

G. Guillot. Inference of structure in subdivided populations at low levels of genetic differentiation. The correlated allele frequencies model revisited. *Bioinformatics*, 24:2222–2228, 2008.

G. Guillot. On the inference of spatial structure from population genetics data. *Bioinformatics*, 25(14):1796–1801, 2009a.

G. Guillot. Response to comment on 'On the inference of spatial structure from population genetics data'. *Bioinformatics*, 25(14): 1805–1806, 2009b.

G. Guillot and M. Foll. Accounting for the ascertainment bias in Markov chain Monte Carlo inferences of population structure. *Bioinformatics*, 25(4):552–554, 2009.

G. Guillot and F. Santos. A computer program to simulate multilocus genotype data with spatially auto-correlated allele frequencies. *Molecular Ecology Resources*, 9(4):1112 – 1120, 2009.

G. Guillot and F. Santos. Using AFLP markers and the Geneland program for the inference of population genetic structure. *Molecular Ecology Resources*, 10(6):1082–1084, 2010.

G. Guillot, A. Estoup, F. Mortier, and J.F. Cosson. A spatial statistical model for landscape genetics. *Genetics*, 170(3):1261–1280, 2005a.

G. Guillot, F. Mortier, and A. Estoup. Geneland: A computer package for landscape genetics. *Molecular Ecology Notes*, 5(3):708–711, 2005b.

G. Guillot, F. Santos, and A. Estoup. Analysing georeferenced population genetics data with Geneland: a new algorithm to deal with null alleles and a friendly graphical user interface. *Bioinformatics*, 24(11):1406–1407, 2008.

G. Guillot, R. Leblois, A. Coulon, and A. Frantz. Statistical methods in spatial genetics. *Molecular Ecology*, 18:4734–4756, 2009.

G. Guillot, S. Renaud, J. Michaux, and J. Claude. A unifying model for the analysis of phenotypic, genetic and geographic data. *To appear in Syst. Biol.*, 2012.

U. Hannelius, E. Salmela, T. Lappalainen, G. Guillot, C.M. Lindgren, U. von Döbeln, P. Lahermo, and J. Kere. Population substructure in Finland and Sweden revealed by a small number of unlinked autosomal SNPs. *BMC Genetics*, 9(54), 2008.

G. Nicholson, A.V. Smith, F. Jónsson, Ó. Gústafsson, K. Stefánsson, and P. Donnelly. Assessing population differentiation and isolation from single-nucleotide polymorphism data. *Journal of the Royal Statistical Society, series B*, 64(4):695–715, 2002.

D. Paetkau, W. Calvert, I. Stirling, and C. Strobeck. Microsatellite analysis of population structure in canadian polar bears. *Molecular Ecology*, 4:347–354, 1995.

E. Paradis. R for beginers. http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf, 2005.

E. Paradis. *Analysis of Phylogenetics and Evolution with R*. Use R. Springer, New York, 2006. URL http://www.springer.com/ 0-387-32914-5. ISBN 0-387-32914-5.

F. Pompanon, A. Bonin, E. Bellemain, and P. Taberlet. Genotyping errors: causes, consequences and solutions. *Nature Review Genetics*, 6(11):847–859, 2005.

J.K. Pritchard, M. Stephens, and P. Donnelly. Inference of population structure using multilocus genotype data. *Genetics*, 155: 945–959, 2000.

F. Rousset. Genepop'007: a complete re-implementation of the Genepop software for windows and linux. *Molecular Ecology Notes*, 8(1):103–106, 2007.

B.S. Weir and C.C. Cockerham. Estimating F-statistics for the analysis of population structure. *Evolution*, 38(6):1358–1370, 1984.

Index