November 22, 2019

BayPass version 2.2

User Manual

Contents

1 Overview

The package BayPass is a population genomics software which is primarily aimed at identifying genetic markers subjected to selection and/or associated to population-specific covariates (e.g., environmental variables, quantitative or categorical phenotypic characteristics). The underlying models explicitly account for (and may estimate) the covariance structure among the population allele frequencies that originates from the shared history of the populations under study. Note that, apart from standard population genetics studies, BayPass is generic enough to be also suited to the analyses of data from other kinds of experiments in which the allele frequency covariance structure is simpler (e.g., experimental evolution). The genetic data typically consists of allele (when derived from individual genotype calls) or read (when derived from Pool–Seq experiments) counts at several markers (for now, BayPass is restricted to the analysis of bi–allelic markers) in several populations. Note that BayPass can handle missing data (no count available in one or several populations) which might be helpful in some contexts.

The core BayPass model is based on the Bayenv model which was proposed by Coop *et al.* (2010) and Günther and Coop (2013). However, as detailed in Gautier (2015), in addition to a complete and independent reprogramming of the core Markov Chain Monte Carlo (MCMC) algorithm, BayPass allows monitoring most of the parameters and the priors of the original models and to introduce various extensions (via e.g., the optional addition of hyper–parameters, the modeling of spatial dependency among consecutive markers).

BayPass is written in Fortran90. The source code and compilation instructions for various platforms (OS X, Windows, Linux) are available. The executable reads data file(s) supplied by the user, and a number of options can be passed through the command line. Some R functions are also provided in the package to facilitate interpretation of the resulting outputs.

This document provides information about how to format the data file, how to specify the user-defined parameters, and how to interpret the results.

2 Before you start

2.1 How to get BayPass?

Download the archive from http://www1.montpellier.inra.fr/CBGP/software/baypass/, and extract it from a terminal:

```
tar -zxvf baypass_2.2.tar.gz
```

2.2 How to compile BayPass?

The source files are to be found in the `src` subdirectory. BayPass is coded in Fortran90 and can therefore be compiled for any system supporting a Fortran90 compiler using the provided `Makefile`. This `Makefile` is designed to work with either the free compiler `gfortran`[1] or the commercial `ifort` intel® Fortran compiler[2]. As a consequence, using another Fortran90 compiler requires modifying the `Makefile` accordingly. Note also that BayPass uses `OpenMP` (`http://openmp.org/wp/`) to implement multi-threading, which allows parallel calculation on computer systems that have multiple CPUs or CPUs with multiple cores. Users thus have to make sure that the corresponding libraries are installed (which is usually the case, at Linux OS or following compiler installation previously described[1]). The following instructions run within the `src` sub-directory allows compiling the code and to produce a binary:

- using the `gfortran` free compiler (the command should automatically produce an executable called `g_baypass`):
  `make clean all FC=gfortran`
- using the `ifort` intel® Fortran compiler (the command should automatically produce an executable called `i_baypass`):
  `make clean all FC=ifort`

A comparison of the computational performances of different compiled versions of the program is given in Table 1. From my own experience, I would recommend using the `ifort` Intel® Fortran compiler instead of `gfortran`, most particularly when running analyses on a single thread. Yet, `gfortran` options may probably be further optimized (any feedback about this is welcome) and the newest versions of the compiler have clearly been improved (`gfortran` $\geq$7.0 is highly recommended). More specifically, `gfortran` executables seem to scale more efficiently than `ifort` ones with increasing number of threads. It should however be noticed that, whatever the compiler used, the speed does not scale linearly with the number of threads and increasing too much the number of threads may even be counter-productive. In general, I would not recommend using more than 8 threads. Nevertheless,

[1] If not already installed in your system, binaries are available at `https://gcc.gnu.org/wiki/GFortranBinaries` and easy to install for most Windows, Mac and Linux OS versions (many thanks to Andrew Beckerman for pointing this web-page to me!)

[2] Intel proposes a free personal non-commercial unsupported version that is essentially identical to the commercial version: `https://software.intel.com/en-us/qualify-for-free-software/academicresearcher`

the performance may strongly depend on the considered options and on the size of the data sets

Compiler	1 thread	4 threads	8 threads	16 threads
`ifort (v16.0.3)`	9 min 50 s	7 min 13 s	6 min 52 s	8 min 03 s
`gfortran (v7.4.0)`	18 min 49 s	8 min 30 s	7 min 10 s	7 min 41 s
`gfortran (v8.3.0)`	17 min 32 s	7 min 57 s	8 min 28 s	8 min 20 s

Table 1: Comparisons of the computational efficiency of different compiled versions of BayPass for the analysis of the Littorina Pool–Seq read count example data set (12 pools, 2,500 SNPs) considered in paragraph 5.2.

In the following, it is assumed that the program was made executable and accessible in your path. For instance, under Linux, this may be achieved by copying the executable in a directory declared in the path (e.g., `/usr/local/bin`) or by adding the program to the `$PATH` system variable (using the `export` command)

Under Linux (or MacOS), before the first use, make sure to give appropriate execution rights to the program. For instance you may run:

```
chmod +x baypass
```

2.3 Input file format

Depending on the type of analyses, different data files might be required by the program. The following examples of the different input files are available in the `examples` directory:

- `geno.bta14`: this file contains allele count data for 18 French cattle breeds at 1,394 SNPs mapping to the BTA14 bovine chromosome (see Gautier (2015) for details).
- `bta.pc1`: this file contains the SMS (Synthetic Morphology Score) for the 18 French cattle breeds (see Gautier (2015) for details).
- `omega.bta`: this file contains the matrix $\mathbf{\Omega}$ for the 18 French cattle breeds ($\widehat{\mathbf{\Omega}}^{\text{bpas}}_{\text{BTA}}$) as estimated under the core model from the whole genome SNP data (see Gautier (2015) for details).
- `lsa.geno`: this file contains read count data (Pool–Seq data) for 12 populations from the *Littorina saxatilis* marine snail (Westram *et al.*, 2014) at 2,500 SNPs randomly chosen among the ones analyzed in Gautier (2015) (but including the ca. 150 outlier SNPs identified).

- `lsa.poolsize`: this file contains the haploid pool sizes of the 12 *Littorina saxatilis* populations.
- `lsa.ecotype`: this file contains the code for the ecotype of the 12 *Littorina saxatilis* populations (-1 for the "crab" habitat and 1 for the "wave" habitat).

Note that for Pool–Seq data, the R package `poolfstat` (Hivert *et al.*, 2018), available from the CRAN repository (`https://cran.r-project.org/web/packages/poolfstat/index.html`), provides functions to generate input files in the appropriate format.

2.3.1 The genotyping data file [always required]

The genotyping data file contains allele or read count (for PoolSeq experiment) data for each of the *nsnp* markers assayed in each of the *npop* sampled populations. The genotyping data file is simply organized as a matrix with *nsnp* rows and $2 * npop$ columns. The row field separator is a space. More precisely, each row corresponds to one marker and the number of columns is twice the number of populations because each pair of numbers corresponds to each allele (or read counts for PoolSeq experiment) counts in one population[3].

As a schematic example, the genotyping data input file for allele count data should read as follows:

```
--- file begins here ---
81 19 86 14 2 98 8 92 32 68 23 77
89 11 81 19 9 91 1 99 27 73 27 73
89 11 91 9 0 0 15 85 77 23 80 20

[...97 more lines...]

--- file ends here ---
```

In this example, there are 6 populations and 100 SNP markers. At the first SNP, in the first population, there are 81 copies of the first allele, and 19 copies of the second allele. In the second population, there are 86 copies of the first allele, and 14 copies of the second allele, etc. Note that both alleles in the third SNP in the third population have 0 copy. This marker will be treated as a missing data in the corresponding population. The file named `geno.bta14` in the `example` directory provides a more realistic example.

Similarly, as a schematic example, the genotyping data input file for allele count data should read as follows:

[3]For now, BayPass is restricted to bi–allelic marker

```
--- file begins here ---
71 8 115 0 61 36 51 39 10 91 69 58
82 0 91 0 84 14 24 57 28 80 18 80
93 28 112 30 0 0 0 113 33 68 0 106

[...97 more lines...]

--- file ends here ---
```

In this example, there are also 6 populations and 100 SNP markers. At the first SNP, in the first population, there are 71 reads of the first allele, and 8 reads of the second allele. In the second population, there are 115 reads of the first allele, and 0 read of the second allele, etc. Note that both alleles in the third SNP in the third population have 0 copie. This marker will be treated as a missing data in the corresponding population. The file named `lsa.geno` in the `example` directory provides a more realistic example.

For Pool–Seq data to be analyzed properly (i.e., not like allele count data), it is necessary to provide a file with the (haploid) size of each pool (see 2.3.2).

2.3.2 The pool (haploid) size file [only required for Pool–Seq data]

For Pool–Seq experiment, the haploid size (twice the number of pooled individuals for diploid species) of each population should be provided. As a schematic example, the pool (haploid) size file should read as follows:

```
--- file begins here ---
60 75 100 90 80 50
--- file ends here ---
```

In this example, there are 6 populations with respective haploid sample sizes of 60 (first population), 75 (second population), 100 (third population), 90 (fourth population), 80 (fifth population) and 50 (sixth population). The order of the populations in the pool size file must be the same as in the allele count (and the covariate) data file(s). The file named `lsa.poolsize` in the `example` directory provides a more realistic example.

2.3.3 The covariate data file [required for the covariate modes]

The values of the covariates (e.g., environmental data, phenotypic traits, etc.) for the different populations should be provided in a file with the format exemplified in the following:

```
--- file begins here ---
150 1500 800 300 200 2500
181.5 172.6 152.3 191.8 154.2 166.8
1 1 0 0 1 1
0.1 0.8 -1.15 1.6 0.02 -0.5
--- file ends here ---
```

In this example, there are 6 populations (columns) and 4 covariates (row). The first covariate might be viewed as a typical environmental covariate, like altitude in meters (the first population is living at ca. 150m above the sea level, the second at ca. 1,500m, and so on); the second as a quantitative trait like average population sizes in cm (individuals from the first population are 181.5 cm tall on average, individuals from the second population 172.6 cm, and so on); the third covariate as a typical binary trait as the presence (1, for the first, second, fifth and sixth populations) or the absence (0, for the third and fourth populations); and the last covariate might be viewed as a synthetic variable like the first principal components of a PCA. The order of the populations (columns) in the covariate data file must be the same as in the allele count (and the pool size) data file(s).

The files named `bta.pc1` and `lsa.ecotype` in the `example` directory provide alternative real-life examples.

Note that in most cases, it is (strongly) recommended to scale each covariate (so that $\hat{\mu} = 0$ and $\widehat{\sigma^2} = 1$ for each covariable). The `scalecov` option allows performing this step automatically prior to analysis, if needed.

2.3.4 The contrast data file [required to compute contrast (under the core model)]

To perform analysis of association with binary traits, one may compute contrast of standardized allele frequencies between two groups of populations (Olazcuaga *et al.*, 2019). The group membership of each population (1 for first group, -1 for the alternative group, and possibly 0 if excluded from the contrast computation) should be provided in a file with the format exemplified in the following:

```
--- file begins here ---
1 -1 1 -1 1 -1 1 -1 1 -1 1 -1
1 -1 0 0 0 0 0 -1 1 -1 1 -1
--- file ends here ---
```

In this example, there are 12 populations (columns) and 2 contrasts (row). The first contrast compare the group of populations #1, #3, #5, #7, #9 and

#11 against the group of populations #2, #4, #6, #8, #10 and #12. The second contrast compare the group of populations #1, #9 and #11 against the group of populations #2, #8, #10 and #12; the populations #3, #4, #5, #6 and #7 being excluded from the comparison.

The file `lsa.ecotype` in the `example` directory provide a real-life example.

2.3.5 The covariance matrix file [optional, required for the AUX covariate mode]

For some applications (see below), it might be interesting (e.g., to parallelize some analyses) or required (when using the AUX covariate mode) to provide the population covariance matrix Ω. As a schematic example, the covariance matrix file reads as follows:

```
--- file begins here ---
0.098053 0.019595 0.032433 -0.029601 -0.024190 -0.029247
0.019595 0.160147 0.018942 -0.027348 -0.039733 -0.039010
0.032433 0.018942 0.149962 -0.054973 -0.058700 -0.057288
-0.029601 0.027348 0.054973 0.187511 0.221914 0.165862
-0.024190 0.039733 0.058700 0.221914 0.562666 0.260231
-0.029247 0.039010 0.057288 0.165862 0.260231 0.219761
--- file ends here ---
```

In this example, there are 6 populations. Hence, the population covariance matrix is a 6×6 squared symmetric matrix. The order of the populations (columns and rows) in the matrix Ω should be the same as the columns in the allele count (and the pool size and the covariate) data file(s). Note that this file is produced in the appropriate format by the program when running BayPass under the core model (see 3.3).

The file named `omega.bta` provides a real-life example.

3 Running BayPass

3.1 Overview of the different models available in BayPass

Directed Acyclic Graphs (DAG) of the different family of models are represented in Figure 1 (see Gautier (2015) for details). Briefly, three types of (closely related) models might be investigated using BayPass, considering either Allele count data (left panel in Figure 1) or Read count data (right panel in Figure 1) as obtained from Pool–Seq experiments.

3.1.1 The core model

The core model depicted in Figure 1A might be viewed as a generalization of the model proposed by Nicholson *et al.* (2002) and was first proposed by Coop *et al.* (2010). This model is the one considered by BayPass when no covariate data file is provided and is actually nested in the others models.

The main parameter of interest is the (scaled) covariance matrix of population allele frequencies $\mathbf{\Omega}$ resulting from their (possibly unknown and complex) shared history. This matrix may also be used for demographic inference. Examples on how to represent $\mathbf{\Omega}$) are provided in section 5.1.1. For instance, $\mathbf{\Omega}$ might be converted (e.g., using the `cov2cor()` function in R stats package) into a correlation matrix $\mathbf{\Sigma}$ further interpreted as a similarity matrix. From this latter matrix, one may define a distance (dissimilarity) matrix (e.g., $d_{ij} = 1- \mid \rho_{ij} \mid$ where d_{ij} is the distance between populations i and j and ρ_{ij} is the element ij of $\mathbf{\Sigma}$) to perform hierarchical clustering[4] and summarize the history of the population as a bifurcating phylogenetic tree (without gene flow). A more complex demographic inference based on an interpretation of the matrix $\mathbf{\Omega}$ (although estimated in a less accurate way) in terms of an admixture graph has been proposed by Pickrell and Pritchard (2012).

The core model allows scanning the genome for differentiation (covariate-free) using the XtX statistics as introduced by Günther and Coop (2013) which is computed by default in BayPass (e.g., see 5.1.1). The main advantage of this approach is to explicitly account for the covariance structure in population allele frequencies (via estimating $\mathbf{\Omega}$) resulting from the demographic history of the populations.

In the current implementation of BayPass, the prior distribution for $\mathbf{\Omega}$ is an Inverse-Wishart: $\mathbf{\Omega} \sim \mathrm{W}_J^{-1}(\rho\mathbf{I}_\mathrm{J}, \rho)$ (where J is the number of populations). By default $\rho = 1$ (rather than $\rho = J$ as in Bayenv) which was found as the most reliable value (Gautier, 2015). Similarly, the hyperparameters a_π and b_π of the prior β distribution for the overall (across population) SNP allele frequencies are estimated by default. However, they might be fixed to $a_\pi = b_\pi = 1$ (as in e.g., Bayenv) using `fixpibetapar` option or to any other values using further the `betapiprior` option (3.2).

3.1.2 The standard covariate model and extensions

The standard covariate model is represented in Figure 1B and is the one considered by default in BayPass when a covariate data file is provided using `-efile` option (3.2). This model allows evaluating to which extent a

[4]For an interesting discussion and examples in R, see http://research.stowers-institute.org/mcm/efg/R/Visualization/cor-cluster/index.htm

population covariable k is (linearly) associated with each marker i (which are assumed independent given $\mathbf{\Omega}$) by the introduction of the regression coefficients β_{ik} (for convenience the indices k for covariables are dropped in Figure 1B).

In the current implementation of BayPass, the prior distribution for the β_{ik}'s is Uniform: $\beta_{ik} \sim \text{Unif}(\beta_{\min}, \beta_{\max})$. By default, $\beta_{\min} = -0.3$ and $\beta_{\max} = 0.3$ but these values might be changed by the user with the `minbeta` and `maxbeta` options respectively (3.2). Note that in Bayenv (Coop *et al.*, 2010), $\beta_{\min} = -0.1$ and $\beta_{\max} = 0.1$.

The estimation of the β_{ik} regression coefficients for each SNP i may be performed using two different approaches (Gautier, 2015):

- Using an Importance Sampling (IS) approximation (default). This also allows estimating Bayes Factor to evaluate the support in favor of association of each SNP i with a covariable k, i.e., to compare the model with association ($\beta_{ik} \neq 0$) against the null model ($\beta_{ik} = 0$). The IS based estimation was initially proposed by Coop *et al.* (2010) and is based on a numerical integration that requires the definition of a grid covering the whole support of the β_{ik} prior distribution. In BayPass, the grid consists of n_β (by default $n_\beta = 201$) equidistant points from $\beta_{\min}$ to $\beta_{\max}$ (including the boundaries) leading to a lag between two successive values equal to $\frac{\beta_{\max}-\beta_{\min}}{n_\beta-1}$ (i.e., 0.003 with default values). Other values for n_β might be supplied by the user with the `-nbetagrid` option (3.2).

- Using an MCMC algorithm (activated via the `covmcmc` option). In this case, the user should provide the matrix $\mathbf{\Omega}$ (e.g., using posterior estimates available from a previous analysis) and it is recommended to consider only one covariable at a time (particularly if some covariables are correlated).

3.1.3 The auxiliary covariate model

The auxiliary covariate model, represented in Figure 1C and activated with the `auxmodel` option, is an extension of the previous model (Figure 1B). It involves the introduction of a Bayesian (binary) auxiliary variable δ_{ik} for each regression coefficient β_{ik} (Gautier, 2015). The auxiliary variable actually indicates whether a specific SNP i can be regarded as associated to a given covariable k ($\delta_{ik} = 1$) or not ($\delta_{ik} = 1$). By looking at the posterior distribution of each auxiliary variable, it is then straightforward to derive a Bayes Factor (BF_{mc}) to compare both models while dealing with multiple testing issues (Gautier, 2015). In addition, the introduction of a Bayesian auxiliary variable makes it easier to account for spatial dependency among markers. In BayPass, the general form of the δ_{ik} prior distribution is indeed that of an 1D Ising model with a parametrization analogous to the one proposed in a similar context by Duforet-Frebourg *et al.* (2014): $\pi\left(\boldsymbol{\delta_k}\right) \propto P^{s_1}(1-P)^{s_0}e^{\eta \mathrm{b_{is}}}$, where $\boldsymbol{\delta_k}$ is the vector of the nsnp auxiliary variables for covariable k, s_1 and s_0 are the number of SNPs associated (i.e. with $\delta_{ik}=1$) and not associated (i.e. with $\delta_{ik}=0$) to the covariable[5], and η corresponds to the number of pairs of consecutive markers (neighbors) that are in the same state at the auxiliary variable[6] (i.e., $\delta_{i,k}=\delta_{i+1,k}$). The parameter P broadly corresponds to the prior proportion of SNP associated to the covariable. In the BayPass auxiliary model, P is assumed a priori beta distributed: $P \sim \beta\left(a_P, b_P\right)$. By default, $a_P = 0.02$ and $b_P = 1.98$ (this values might be changed by the user with the `-auxPbetaprior` option) which amounts to consider that only a small fraction of the SNPs ($\frac{a_P}{a_P+b_P}$=1%) are a priori expected to be associated to the covariable while allowing some uncertainty on this key parameter (e.g., the prior probability of P >10% being equal to 0.028 with default parameters). The parameter $\mathrm{b_{is}}$, called the inverse temperature in the Ising (and Potts) model literature, determines the level of spatial homogeneity of the auxiliary variables between neighbors. In BayPass, $\mathrm{b_{is}}=0$ by default implying that auxiliary variables are independent (no spatial dependency). Note that $\mathrm{b_{is}}=0$ amounts to assume the δ_{ik} follows a Bernoulli distribution with parameter P. Conversely, $\mathrm{b_{is}}>0$ leads to assume that the δ_{ik} with similar values tend to cluster according to the underlying SNP positions (the higher the $\mathrm{b_{is}}$, the higher the level of spatial homogeneity). In biological terms, SNP associated to a given covariable might be expected to cluster due to Linkage Disequilibrium with the underlying (possibly not genotyped) causal variant(s). In practice, $\mathrm{b_{is}}=1$ is commonly used and a value of $\mathrm{b_{is}} \leq 1$ is recommended.

In technical terms, the overall parametrization of the Ising prior assumes no external field and no weight (as in the so-called compound Ising model) between the neighboring auxiliary variables. In the current implementation of the BayPass auxiliary covariate model (when $\mathrm{b_{is}}>0$), the information about between SNPs distances is therefore not accounted for. Only the relative position of markers are considered. For the applications where this modeling might be relevant (whole genome scan), this corresponds to assuming a relative homogeneity in marker spacing as measured by genetic (rather than physical) distances (which might be unavailable, in practice).

[5] $s_1 = \sum\limits_{i=1}^{\mathrm{nsnp}} \delta_{ik}=1$ and $s_0 = \sum\limits_{i=1}^{\mathrm{nsnp}} \delta_{ik}=0$

[6] $\eta = \sum\limits_{i\sim j} \mathbf{1}_{\delta_{ik}=\delta_{jk}}$

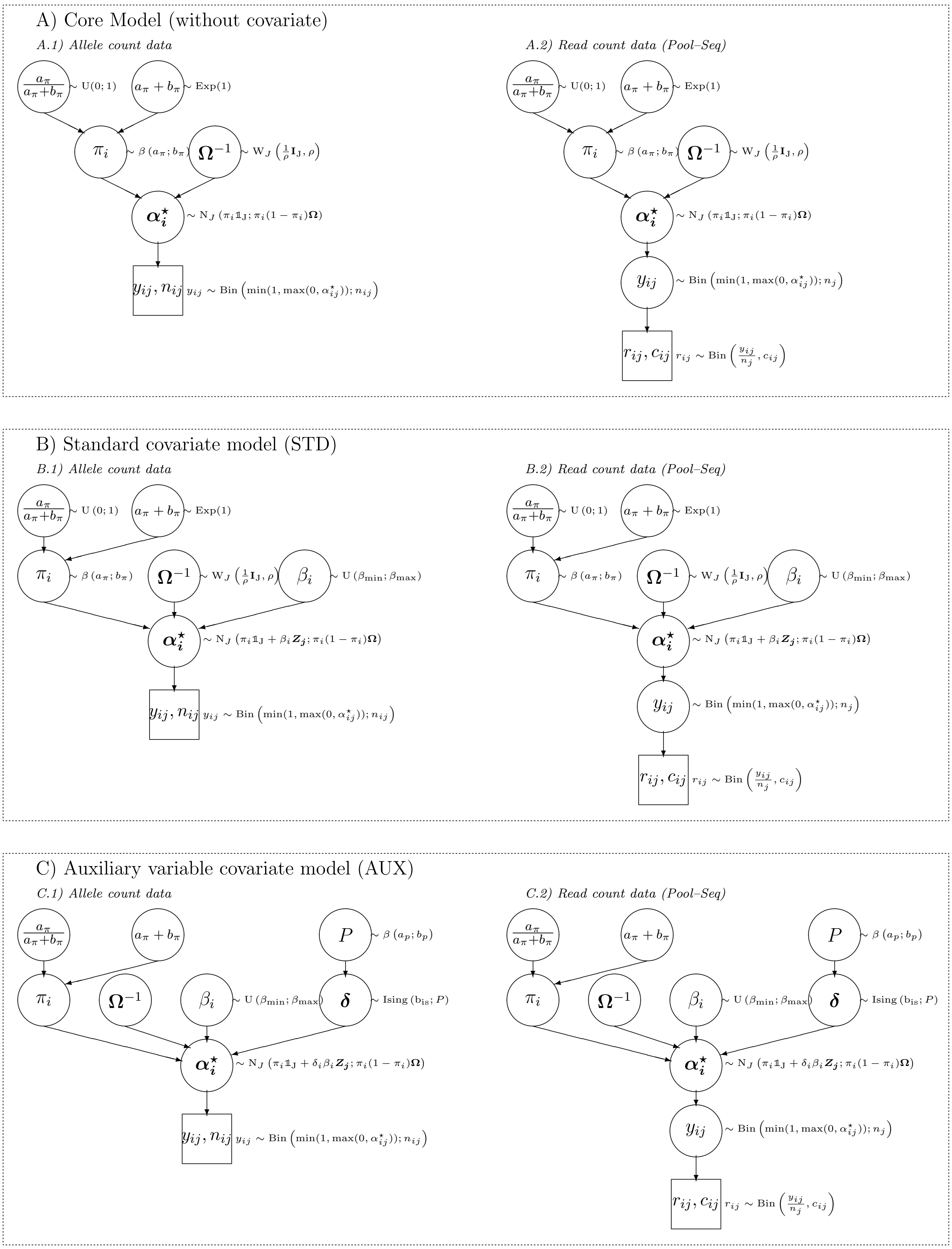
A) Core Model (without covariate)
A.1) Allele count data
A.2) Read count data (Pool–Seq)
B) Standard covariate model (STD)
B.1) Allele count data
B.2) Read count data (Pool–Seq)
C) Auxiliary variable covariate model (AUX)
C.1) Allele count data
C.2) Read count data (Pool–Seq)
$\frac{a_\pi}{a_\pi+b_\pi} \sim \mathrm{U}(0;1)$
$a_\pi + b_\pi \sim \mathrm{Exp}(1)$
$\pi_i \sim \beta(a_\pi; b_\pi)$
$\boldsymbol{\Omega}^{-1} \sim \mathrm{W}_J\left(\frac{1}{\rho}\mathbf{I}_J, \rho\right)$
$\beta_i \sim \mathrm{U}(\beta_{\min}; \beta_{\max})$
$P \sim \beta(a_p; b_p)$
$\boldsymbol{\delta} \sim \mathrm{Ising}(\mathrm{b_{is}}; P)$
$\boldsymbol{\alpha_i^\star} \sim \mathrm{N}_J(\pi_i \mathbb{1}_J; \pi_i(1-\pi_i)\boldsymbol{\Omega})$
$\boldsymbol{\alpha_i^\star} \sim \mathrm{N}_J(\pi_i \mathbb{1}_J + \beta_i \boldsymbol{Z_j}; \pi_i(1-\pi_i)\boldsymbol{\Omega})$
$\boldsymbol{\alpha_i^\star} \sim \mathrm{N}_J(\pi_i \mathbb{1}_J + \delta_i\beta_i \boldsymbol{Z_j}; \pi_i(1-\pi_i)\boldsymbol{\Omega})$
y_{ij}, n_{ij} $y_{ij} \sim \mathrm{Bin}\left(\min(1, \max(0, \alpha_{ij}^\star)); n_{ij}\right)$
$y_{ij} \sim \mathrm{Bin}\left(\min(1, \max(0, \alpha_{ij}^\star)); n_j\right)$
r_{ij}, c_{ij} $r_{ij} \sim \mathrm{Bin}\left(\frac{y_{ij}}{n_j}, c_{ij}\right)$

Figure 1: Directed Acyclic Graphs of the different hierarchical Bayesian models available in BayPass (see 3.1).

3.2 Detailed overview of all the options

BayPass is a command-line executable. The ASCII hyphen-minus ("-") is used to specify options. As specified below, some options take integer or float values and some options do not. Here is an example call of the program:

```
baypass -gfile data.geno -efile env.data -outprefix ana1
```

The full list of the options accepted by BayPass is printed out using the command: `baypass -help` as follows:

```
Version 2.2

Usage: BayPass [options]

Options:
I)   General Options:
 -help                   Display the help page
 -gfile           CHAR   Genotyping Data File                                        (always required)
 -efile           CHAR   Covariate file: activate Covariate Mode                     (def="")
 -scalecov        CHAR   Scale covariates                                            (def="")
 -contrastfile    CHAR   Contrast to be computed                                     (def="")
 -poolsizefile    CHAR   Name of the Pool Size file => activate PoolSeq mode         (def="")
 -outprefix       CHAR   Prefix used for the output files                            (def="")

II)  Model Options:
 -omegafile       CHAR   Omega matrix file => inactivate estim. of omega             (def="")
 -rho             INT    Rho parameter of the Wishart prior on omega                 (def=1)
 -setpibetapar           Inactivate estimation of the Pi beta priors parameters
 -betapiprior     FLOAT2 Pi Beta prior parameters (if -setpibetapar)                 (def=1.0 1.0)
 -minbeta         FLOAT  Lower beta coef. for the grid                               (def=-0.3)
 -maxbeta         FLOAT  Upper beta coef. for the grid                               (def= 0.3)

 I.1)  IS covariate mode (default covariate mode):
  -nbetagrid       INT    Number of grid points (IS covariate mode)                 (def=201)

 I.2)  MCMC covariate mode:
  -covmcmc                Activate mcmc covariate mode (desactivate estim. of omega)
  -auxmodel               Activate Auxiliary variable mode to estimate BF
  -isingbeta       FLOAT  Beta (so-called inverse temperature) of the Ising model       (def=0.0)
  -auxPbetaprior   FLOAT2 auxiliary P Beta prior parameters                         (def=0.02 1.98)

III) MCMC Options:
 -nthreads        INT    Number of threads                                           (def=1)
 -nval            INT    Number of post-burnin and thinned samples to generate       (def=1000)
 -thin            INT    Size of the thinning (record one every thin post-burnin sample) (def=20)
 -burnin          INT    Burn-in length                                              (def=5000)
 -npilot          INT    Number of pilot runs (to adjust proposal distributions)     (def=20)
 -pilotlength     INT    Pilot run length                                            (def=500)
 -accinf          FLOAT  Lower target acceptance rate bound                          (def=0.25)
 -accsup          FLOAT  Upper target acceptance rate bound                          (def=0.40)
 -adjrate         FLOAT  Adjustement factor                                          (def=1.25)
 -d0pi            FLOAT  Initial delta for the pi all. freq. proposal                (def=0.5)
 -upalphaalt             Alternative update of the pij
 -d0pij           FLOAT  Initial delta for the pij all. freq. proposal (alt. update)     (def=0.05)
 -d0yij           INT    Initial delta for the yij all. count (PoolSeq mode)         (def=1)
 -d0cj            FLOAT  If nicholsonprior is set for Omega, initial delta for the cj    (def=0.05)
 -seed            INT    Random Number Generator seed                                (def=5001)
 -print_omega_samples    Print post-burnin and thinned samples of Omega in a file
```

In this menu, each option is followed by i) the kind of argument (if any) required (e.g., INT for integer, FLOAT for real, FLOAT2 for a pair of space separated real numbers); ii) a brief description of its function; and iii) the default value.

In the following, we detailed all the options of BayPass:

`-help`

This option prints out the help menu (see above). Note that this option is dominating all the other options, i.e. if `-help` is used in conjunction with any other option of the program, the help menu is displayed. No argument is required for this option.

`-gfile`

This option (mandatory) gives the name of the genotyping input file. See 2.3.1 for a description of the corresponding input file format. The required argument must be a character string (name of the input file) without space (e.g., `-gfile data.geno` if the input file is named "data.geno").

`-efile`

This option gives the name of the covariate input file. See 2.3.3 for a description of the corresponding input file format. The required argument must be a character string (name of the input file) without space (e.g., `-efile data.env` if the input file is named "data.env").

`-scalecov`

This option allows scaling each covariable in the covariate input file (See 2.3.3). No argument is required for this option. If activated, an output file named "`covariate.std`" containing the scaled covariables is produced.

`-contrastfile`

This option gives the name of the covariate input file. See 2.3.4 for a description of the corresponding input file format. The required argument must be a character string (name of the input file) without space (e.g., `-contrastfile data.contrast` if the input file is named "data.env").

`-poolsizefile`

This option gives the name of the input file containing the haploid sample size of each population. See 2.3.2 for a description of the corresponding input file format. The required argument must be a character string (name of the input file) with no space (e.g., `-poolsizefile data.poolsize` if the input file is named "data.poolsize"). Note that this option automatically activates the Pool–Seq mode, i.e., the PoolSeq version of the different models are considered (as represented in Figures 1A2, B2 and C2).

`-outprefix`

This option allows adding a prefix to all the output files. The required argument must be a character chain without space. For instance, if using `-outprefix ana1`, the name of all the output files will begin by "`ana1_`". By default, no prefix is added.

`-omegafile`

This option gives the name of the input file for the population covariance matrix ($\mathbf{\Omega}$ in 3.1.1 and Figure 1). See 2.3.5 for a description of the corresponding input file format. The required argument must be a character string (name of the input file) with no space (e.g., `-omegafile matrix.dat` if the input file is named "matrix.dat"). This option inactivates the estimation of $\mathbf{\Omega}$ and is mandatory in the covariate models involving estimation of the regression coefficients via MCMC, i.e., both the standard model (see 3.1.2) with the `-covmcmc` option and the auxiliary variable model with the `-auxmodel` option (see 3.1.3).

`-rho`

This option allows specifying the value of ρ for the Inverse-Wishart prior of $\mathbf{\Omega}$ (see Figure 1 and 3.1.1). The required argument must be a positive integer. By default, `-rho 1` (i.e., $\rho = 1$).

`-setpibetapar`

This option allows inactivating the estimation of the two (hyper–)parameters a_π and b_π of the prior β distribution for the overall (across population) SNP allele frequencies (see Figure 1 and 3.1.1) (and set them to the values specified with the `-betapiprior` option). No argument is required for this option.

`-betapiprior`

This option allows specifying the values of the two (hyper–)parameters a_π and b_π (respectively) of the prior β distribution for the overall (across population) SNP allele frequencies (see Figure 1 and 3.1.1). The required argument must be two positive real numbers. By default `-betapiprior 1.0 1.0` (i.e., $a_\pi = b_\pi = 1$).

`-minbeta`

This option allows specifying the lower bound of the Uniform prior distribution on the regression coefficients (see Figure 1 and 3.1.2). The required argument must be a real number (lower than `maxbeta` defined below). By default `-minbeta -0.3` (i.e., $\beta_{\min} = -0.3$).

`-maxbeta`

This option allows specifying the upper bound of the Uniform prior distribution on the regression coefficients (see Figure 1 and 3.1.2). The required argument must be a real number (greater than `minbeta` defined above). By default `-maxbeta 0.3` (i.e., $\beta_{\max} = 0.3$).

`-nthreads`

This option gives the number of threads to be used for parallel computations. By default, `-nthreads 1` (i.e., a single core is used hence no parallelization).

`-nval`

This option gives the number of post–burn–in (and thinned MCMC) samples recorded from the posterior distributions of the parameters of interest. The required argument must be a positive integer. By default, `-nval 1000` (i.e., 1,000 post burn-in and thinned samples are generated). Note that with default values, the total number of iterations of the MCMC sampler run after the burn-in period is equal to 25,000 (since by default, the thinning rate is equal to 25, see `-thin` option)

`-thin`

This option gives the size of the thinning (i.e., the number of iterations between any two records from the MCMC). The required argument must be a positive integer. By default, `-thin 25` (i.e., the size of the thinning is 25).

`-burnin`

This option gives the length of the burn-in period (i.e., the number of iterations before the first record from the MCMC). The required argument must be a positive integer. By default, `-burnin 5000` (i.e., 5,000 iterations are run during the burn–in period).

`-npilot`

This option gives the number of pilot runs (i.e., the number of runs used to adjust the parameters of the MCMC proposal distributions of parameters updated through a Metropolis-Hastings algorithm). The targeted acceptance rates are defined with the `-accinf` and `-accsup` options (by default, these are set to 0.25 and 0.40 respectively). The required argument must be a positive integer. By default, `-npilot 20` (i.e., 20 pilot runs are performed).

`-pilotlength`

This option gives the number of iterations of each pilot run (see `npilot` option above). The required argument must be a positive integer. By default, `-pilotlength 500` (i.e., each pilot run consist of 500 iterations).

`-accinf`

This option gives the lower bound of the targeted acceptance rates to adjust the parameters of the MCMC proposal distributions of parameters updated through a Metropolis-Hastings algorithm, during the pilot runs. For instance, in the case of a uniform proposal distribution of the form Unif($x - \delta, x + \delta$) (where x represents the current value of the parameter of interest and δ specifies the size of the support), if acceptance rates are below this lower bound after a pilot run, then δ is increased (e.g., multiplied by a factor defined with the `adjrate` parameter and set to 1.25 by default). The required argument must be a positive real number (< 1 and lower than `accsup` defined below). By default, `-accinf 0.25` (i.e., acceptance rates should be at least equal to 25%).

`-accsup`

This option gives the upper bound of the targeted acceptance rates to adjust the parameters of the MCMC proposal distributions of parameters updated through a Metropolis-Hastings algorithm, during the pilot runs. For instance, in the case of a uniform proposal distribution of the form Unif($x - \delta, x + \delta$) (where x represents the current value of the parameter of interest and δ specifies the size of the support), if acceptance rates are above this upper bound after a pilot run, then δ is decreased (e.g., divided by a factor defined with the `adjrate` parameter and set to 1.25 by default). The required argument must be a positive real number (≤ 1 and greater than `accinf` defined above). By default, `-accsup 0.40` (i.e., acceptance rates should be less than 40%).

`-adjrate`

This option gives the factor used to adjust the parameters of the MCMC proposal distributions of parameters updated through a Metropolis-Hastings algorithm, during the pilot runs. For instance, in the case of a uniform proposal distribution of the form Unif$(x-\delta, x+\delta)$ (where x represents the current value of the parameter of interest and δ specifies the size of the support), if acceptance rates are below (respectively above) the lower (respectively upper) bound of the targeted regions (as defined above with the `-accinf` and `-accsup` options) after a pilot run, then δ is multiplied (respectively, divided) by this factor. The required argument must be a real number > 1. By default, `-adjrate 1.25`.

`-d0pi`

This option gives the initial value of the δ_π, which is half the window width from which proposal values of the overall SNP allele frequencies π_i (see Figure 1) are drawn uniformly around the current value π_i in the Metroplis–Hastings update. The value of δ_π is eventually adjusted for each locus during the pilot runs (see options `-npilot`, `-pilotlength`, `-accinf`, `-accsup` and `-adjrate`). The required argument must be a positive real number. By default, `-d0pi 0.5` (i.e., $\delta_p = 0.5$).

`-upalphaalt`

This option activates an alternative Metropolis–Hastings algorithm for the population SNP allele frequencies α_{ij} (see Figure 1). By default, the proposal is the same as the one described by Coop *et al.* (2010) (Appendix A). Briefly, denoting $\boldsymbol{\alpha_{i.}}$ as the vector of allele frequencies for SNP i in each population, the vector $\boldsymbol{\alpha^{\star}}_{\boldsymbol{i}}^{cdt}$ evaluated in a given Metropolis–Hastings update is sampled from the following multivariate Gaussian distribution: $\boldsymbol{\alpha^{\star}}_{\boldsymbol{i}}^{cdt} \sim MNV\left(\boldsymbol{\alpha^{\star}}_{\boldsymbol{i}}, \boldsymbol{\Gamma}\sigma_i^2\right)$ where $\boldsymbol{\Gamma}$ is obtained by a Choleski decomposition of the matrix $\boldsymbol{\Omega}$ (i.e., $\boldsymbol{\Omega} = {}^{\boldsymbol{t}}\boldsymbol{\Gamma}\boldsymbol{\Gamma}$). The alternative proposal activated with the `-upalphaalt` option is defined on a SNP by population basis and is a uniform distribution centered on the current values of the parameters (i.e., ${\alpha^{\star}}_{ij}^{cdt} \sim$ Unif$({\alpha^{\star}}_{ij} - \lambda_{ij}^{\alpha}, \alpha_{ij}^{\star} - \lambda_{ij}^{\alpha})$). The algorithm is slower than the default one but may perform better, in particular when sample sizes are heterogeneous across samples. No argument is required for this option.

`-d0pij`

This option gives the initial value of the δ_α used in the proposal distribution of the population SNP allele frequencies α_{ij} in the Metroplis–Hastings updates. Following the notations used above (see `-upalphaalt` option), $\delta_\alpha = \sigma_i^2$ for the default algorithm and $\delta_\alpha = \lambda_{ij}^\alpha$ for the alternative algorithm. The value of δ_α is adjusted for each locus (and population, in the case of the alternative algorithm) during the pilot runs (see options `-npilot`, `-pilotlength`, `-accinf`, `-accsup` and `-adjrate`). The required argument must be a positive real number. By default, `-d0pij 0.05` (i.e., $\sigma_i^2 = 0.05$ for the default algorithm and $\lambda_{ij}^\alpha = 0.05$ for the alternative algorithm).

`-d0yij`

This option gives, in the Pool–Seq mode, the initial value of the δ_y used in the proposal distribution of the population SNP allele count Y_{ij} in the Metroplis–Hastings updates. The value of δ_y is eventually adjusted for each locus and each population during the pilot runs (see options `-npilot`, `-pilotlength`, `-accinf`, `-accsup` and `-adjrate`). The required argument must be a positive integer number lower than the haploid pool sizes. By default, `-d0yij 1` (i.e., $\delta_y = 1$).

`-seed`

This option gives the initial seed of the (pseudo-)Random Number Generator. The required argument must be a positive integer number. By default, `-seed 5001`.

`-print_omega_sample`

This option allows printing the *nval* (as defined with option `-nval`) post-burnin and thinned MCMC samples of the matrix $\mathbf{\Omega}$. If activated, an output file with suffix “`omegasamples.out`” is produced.

3.3 Format of the output files

While running, BayPass prints on the console information regarding the execution of the analysis (these might be redirected in a log file using the `> log.file` unix syntax). At the end of the analysis BayPass produces several output files which may vary according to the considered options (see 3.2). In addition, the name of these different output files may be preceded by the prefix defined with the `-outprefix` option (see 3.2).

In the following, all the output files that may be generated by BayPass are detailed:

- *(outprefix_)*`summary_pij.out` (default mode; e.g., for allele count data) or *(outprefix_)*`summary_yij_pij.out` (Pool–Seq mode; e.g., for read count data)

  These files contain for each locus (`MRK` column) within each population (`POP` column), the mean (`M_P` column) and the standard deviation (`SD_P` column) of the posterior distribution of the $\alpha^{\star}_{ij}$ parameter (see Figure 1) that is closely related to the frequency of the reference allele[7] except that its support is on the real line (hence possible values < 0 or > 1). It also contains the posterior mean (`M_Pstd` column) and the standard deviation (`SD_Pstd` column) of the standardized allele frequency α^{std} ($\boldsymbol{\alpha}^{\text{std}} = \boldsymbol{\Gamma}^{-1}\boldsymbol{\alpha}^{\star}$). In the Pool–Seq mode (i.e., in the *(outprefix_)*`summary_yij_pij.out` file), the columns `M_Y` and `SD_Y` report the posterior means and the posterior standard-deviations of allele counts of each SNP within each population.

- *(outprefix_)*`summary_pi_xtx.out`

  This file contains for each locus (`MRK` column), the mean (`M_P` column) and the standard deviation (`SD_P` column) of the posterior distribution of the (across populations) frequency π_i of the SNP reference allele (see Figure 1). In addition, this file contains for each SNP, the posterior mean (`M_XtX` column) and standard deviation (`SD_XtX` column) of the XtX statistics introduced by Günther and Coop (2013) to identify outlier loci in genome–scan of adaptive differentiation (see 3.1.1). The last two columns (named `XtXst` and `log10(1/pval)`) respectively contains the $XtX^{\star}$ calibrated estimator of the XtX statistic and its corresponding p–value (on a $-\log_{10}$ scale) assuming a χ^2 distribution with *npop* degrees of freedom (Olazcuaga *et al.*, 2019). It should be noticed that these p–values are computed bilaterally[8] to allow the identification of SNPs under either balancing (unexpected low XtX value) or positive (unexpected high XtX value) selection.

[7] $\alpha_{ij} = 1 \wedge (0 \vee \alpha^{\star}_{ij})$

[8] as $p = 1 - 2 \mid \Phi_{\chi^2_J}\left(\widehat{\text{XtX}}\right) - 0.5 \mid$ where $\Phi_{\chi^2(J)}$ represents the cumulative density function of the χ^2 distribution with J degrees of freedom. If computed unilaterally as $p = 1 - \Phi_{\chi^2_J}\left(\widehat{\text{XtX}}\right)$ (resp. $p = \Phi_{\chi^2_J}\left(\widehat{\text{XtX}}\right)$) to detect SNPs subjected to positive (resp. balancing) selection, the presence of the two modes of selection would indeed result in a (not well behaved) U-shaped p–value distribution.

- *(outprefix_)*`summary_lda_omega.out`

  This file contains the posterior means and posterior standard deviations of each element of the *npop* × *npop* scaled population allele frequencies covariance matrix $\mathbf{\Omega}$ (`M_omega_ij` and `SD_omega_ij` columns respectively) as described in Figure 1 (see also 3.1.1), and its inverse $\mathbf{\Lambda} = \mathbf{\Omega}^{-1}$ (`M_lambda_ij` and `SD_lambda_ij` columns respectively).

- *(outprefix_)*`mat_omega.out`

  This file contains the posterior means of the elements of $\mathbf{\Omega}$ in a matrix format. Note that this file is in the format required by the `-omegafile` option of BayPass.

- *(outprefix_)*`omegasamples.out` (if the `-print_omega_samples` option is activated)

  This file contains the *nval* post-burnin and thinned MCMC samples of the matrix $\mathbf{\Omega}$. The $\mathbf{\Omega}$ samples are printed one after the other (i.e., the file has *npop* × *nval* rows and *npop* columns)

- *(outprefix_)*`summary_beta_params.out`

  This file contains the posterior mean (`Mean` column) and standard deviation (`SD` column) of the two parameters (a_π and b_π) of the Beta prior distribution assumed for the (across populations) frequencies of the SNP reference allele (see Figure 1).

- *(outprefix_)*`summary_contrast.out` (if the `contrastfile` options is activated)

  This file contains for each locus (`MRK` column), the posterior mean (`M_C2` column) and the standard deviation (`SD_C2` column) of the C_2 contrast statistic. The last two columns (named `C2_std` and `log10(1/pval)`) respectively contains the (more useful in practice) calibrated estimator of the C_2 statistic and its corresponding p–value (on a $-\log_{10}$ scale) assuming a χ^2 distribution with 1 degree of freedom (Olazcuaga *et al.*, 2019).

- *(outprefix_)*`summary_betai_reg.out`

  This file is only produced the standard covariate mode (see Figure 1B and 3.1.2) where the Importance Sampling algorithm is used to estimate the Bayes Factor (column `BF(dB)`) in dB units (i.e., $10 \times \log_{10}(\text{BF})$) measuring the support of the association of each SNP with each population covariable and the corresponding regression coefficients β_i (column `Beta_is`). T file also contains the empirical Bayesian P–value (`eBPis`) in the $\log_{10}$ scale i.e. $\text{eBPis} = -\log_{10}(1 - 2 \mid 0.5 - \Phi(\widehat{\mu_\beta}/\widehat{\sigma_\beta}) \mid)$ (where $\Phi(x)$ represents the cumulative distribution function for the standard normal distribution) and thus allowing to evaluate the support in favor of a non-null regression coefficient (e.g., eBPis > 3). This file contains for each covariable and each SNP, the posterior mean and standard deviation of the Pearson correlation coefficient (columns `M_Pearson` and `SD_Pearson` respectively) between the scaled allele frequencies $\widetilde{\boldsymbol{\alpha}_{\boldsymbol{i}}^{\star}} = \left\{ \frac{\alpha_{ij}^{\star} - \pi_i}{\sqrt{\pi(1-\pi_i)}} \right\}_{(1..J)}$ and the given covariable after rotation of both vectors by $\boldsymbol{\Gamma}^{-1}$ (see Günther and Coop, 2013) where $\boldsymbol{\Gamma}$ is obtained by a Choleski decomposition of the matrix $\boldsymbol{\Omega}$ (i.e., $\boldsymbol{\Omega} = {}^{t}\boldsymbol{\Gamma}\boldsymbol{\Gamma}$).

- *(outprefix_)*`summary_betai.out` (generated with the `-covmcmc` option)

  This file is produced in place of the *(outprefix_)*`summary_betai_reg.out` described above when the `-covmcmc` option is activated (see 3.1.2). Under the standard model (default), the file contains for each SNP, the posterior mean $\widehat{\mu_\beta}$ (`M_Beta` column) and the standard deviation $\widehat{\sigma_\beta}$ (`SD_Beta` column) of the regression coefficient β_i together with the adjusted δ_β parameter (`DeltaB` column) of the proposal distribution and the post-burn–in acceptance rate (`AccRateB` column). The file also contains an approximate Bayesian P–value (`eBPmc`)[9] to evaluate the support for a non-null regression coefficient (e.g., eBPmc > 3). Under the model with auxiliary variables (`-auxmodel` option, see 3.1.3), the file contains for each SNP, the posterior mean (`M_Beta` column) and the standard deviation (`SD_Beta` column) of the regression coefficient β_i; the posterior mean of the auxiliary variable (column `PIP`[10]); and the estimated Bayes Factor (column `BF(dB)`) in dB units (i.e., $10\times\log_{10}(\text{BF})$) comparing the models with ($\beta_i \neq 0$) and without ($\beta_i = 0$) association of the SNP with the given covariable.

- *(outprefix_)*`summary_Pdelta.out` (covariate model with auxiliary variable, i.e. `-auxmodel` option, see 3.1.3)

[9] $\text{eBPmc} = -\log_{10}(1 - 2 \mid 0.5 - \Phi(\widehat{\mu_\beta}/\widehat{\sigma_\beta}) \mid)$ where $\Phi(x)$ represents the cumulative distribution function for the standard normal distribution

[10] In the model averaging literature, the posterior mean of δ_i actually corresponds to the Posterior Inclusion Probability of the SNP i

This file contains the posterior mean (`M_P` column) and the standard deviation (`SD_P` column) of the parameter P (see Figure 1C and 3.1.3) corresponding to the overall proportion of SNPs associated with each given covariable.

- *(outprefix_)*`covariate.std` (generated by the `-scalecov` option)

  This file contains the scaled covariables.

- *(outprefix_)*`DIC.out`

  This files contains the average deviance (`bar(D)` column), the effective number of parameters of the models (`pD` column) and the Deviance Information Criterion (`DIC` column) as defined in Spiegelhalter *et al.* (2002) and that might be relevant for model comparison purposes. In addition, the logarithm of the pseudo-marginal likelihood of the model is also provided (`LPML` column).

4 Miscellaneous R functions

The `baypass_utils.R` file in the `utils` directory contains six R functions (R Core Team, 2015) (`simulate.baypass()`; `plot.omega()`; `fmd.dist()`, `geno2YN()` and `simulate.PCcorrelated.covariate()`) that may be helpful to interpret some of the results obtained with BayPass. To use these functions, one may simply need to source the corresponding files and ensure that the packages `mvtnorm` (Genz *et al.*, 2015) and `geigen` (Hasselman, 2015) are installed. Although not required by theses functions, the packages `corrplot` (Wei, 2013) and `ape` (Paradis *et al.*, 2004) may be useful for the visualization of the $\mathbf{\Omega}$ matrix (see 5).

4.1 The R function *simulate.baypass()*

4.1.1 Description

The R function `simulate.baypass()` allows simulating either allele or read count data under the core inference model (Figure 1A) and possibly under the STD covariate model (Figure 1B). It produces several objects and output files in a format directly appropriate for analyses with BayPass and Bayenv2[11]. In practice, this function is useful to generate POD for calibration of the XtX

[11] For analyses with Bayenv2, make sure fixed loci have been removed, i.e., `remove.fixed.loci=TRUE`

differentiation measure (or any other measures). More broadly, because the $\mathbf{\Omega}$ matrix capture the demographic history of the populations, this function might also be viewed as a simulator of population genetics data.

4.1.2 Usage

```
simulate.baypass(omega.mat,nsnp=1000,beta.coef=NA,beta.pi=c(1,1),
                 pop.trait=0,sample.size=100,pi.maf=0.05,suffix="sim",
                 remove.fixed.loci=FALSE,coverage=NA)
```

4.1.3 Arguments (in alphabetic order)

- `beta.pi` (def=c(1,1))

  A vector with two elements giving the parameters a_π and b_π respectively, for the Beta distribution of the π_i ("ancestral") allele frequencies.

- `beta.coef` (def=NA; required for simulation under the STD covariate model)

  A vector giving the values of the regression coefficients (β_i in Figure 1) for the simulated associated SNPs (the number of the simulated associated SNPs is equal to the dimension of the vector).

- `coverage` (def=NA; required to activate simulation of read count data)

  Either a single value or a matrix giving the total read counts. In the latter case, the vector of total read counts for each simulated SNP are sampled with replacement from the row of the matrix. The number of columns of the matrix must equal the number of populations, but no restriction are set for the number of rows. For instance, if the matrix has only one row, all the SNPs will have the same read counts within a given population.

- `omega.mat` (always required)

  A positive definite and symmetric matrix of rank *npop* corresponding to the covariance matrix of population allele frequencies ($\mathbf{\Omega}$ in Figure 1)

- `nsnp` (def=1000)

  A single number giving the number of neutral SNPs to simulate.

- `pi.maf` (def=0.05)

  A single value giving the MAF threshold on the simulated π_i ("ancestral") allele frequencies. In the simulation procedure, the π_i's are sampled from the Beta distribution with parameters specified by the `beta.pi` argument. For a given SNP i, if $\pi_i <$`pi.maf` (resp. $\pi_i >$ 1−`pi.maf`) then π_i is set equal to `pi.maf` (resp. 1−`pi.maf`). Setting `pi.maf`= 0 inactivates MAF filtering.

- `pop.trait` (def=0; required for simulation under the STD covariate model)

  A vector of length *npop* giving each population-specific covariable values (the ordering of the populations is assumed to be the same as in the `omega.mat` matrix). By default all values are set to 0 (meaning that the associated SNPs behave neutrally irrespective of their values at the regression coefficients).

- `remove.fixed.loci` (def=FALSE)

  A logical indicating wether or not the monomorphic SNPs (in the observed simulated data) should be discarded.

- `sample.size` (def=100)

  If simulating allele count data, either a single value or a matrix giving the total allele counts (e.g., twice the number of genotyped individuals for autosomal SNPs in a diploid species). In the latter case, the vector of total allele counts for each simulated SNP are sampled with replacement from the matrix rows. The number of columns of the matrix must equal the number of populations, but there is no restriction for the number of rows. For instance, if the matrix has only one row, all the SNPs will have the same allele counts within a given population.

  If simulating read count data, either a single value or a vector of length *npop* giving the pool haploid sample sizes of each population.

- `suffix` (def="sim")

  A character string giving the suffix of the output files generated by the function.

4.1.4 Values

The function produces a list with the following components:

- `omega.sim`

  The matrix used for simulations (declared with `omega.mat`)

- `alpha.sim`

  A matrix with *nsnp* rows and *npop* columns giving the (unbounded) allele frequencies for each simulated SNPs within each population (i.e., $\alpha^{\star}_{ij}$ in Figure 1).

- `pi.sim`

  A vector of length *nsnp* giving the simulated "ancestral" allele frequencies of each SNP (i.e., π_i in Figure 1).

- `N.sim`

  A matrix with *nsnp* rows and *npop* columns giving the total allele counts for each simulated SNP within each population.

- `Y.sim`

  A matrix with *nsnp* rows and *npop* columns giving the allele counts for the reference allele for each simulated SNP within each population.

- `N.pool` (read count data only)

  A matrix with *nsnp* rows and *npop* columns giving the total read counts for each simulated SNP within each population.

- `Y.pool` (read count data only)

  A matrix with *nsnp* rows and *npop* columns giving the read counts for the reference allele for each simulated SNP within each population.

- `betacoef.sim` (simulation under the STD covariate model only)

  A vector of length *nsnp* giving the regression coefficients of each simulated SNP.

In addition, the following output files are printed out (the extension `.suffix` is the one defined with the `suffix` argument):

- `G.`*suffix*

  The allele count data file in BayPass format (see 2.3).

- `Gpool.`*suffix* (when simulating read count data)

  The read count data file in BayPass format (see 2.3).

- `bayenv_freq.`*suffix*

  The allele count data file in Bayenv2 format[11].

- `bayenv_freq_pool.`*suffix* (when simulating read count data)

  The read count data file in Bayenv2 format[11].

- `alpha.`*suffix*

  The matrix of (unbounded) allele frequencies (*nsnp* rows and *npop* columns) for each simulated SNP within each population (i.e., $\alpha^{\star}_{ij}$ in Figure 1).

- `pi.`*suffix*

  The vector of simulated "ancestral" allele frequencies for each simulated SNP (i.e., π_i in Figure 1).

- `betacoef.`*suffix* (when simulating under the STD covariate model)

  The regression coefficients of each simulated SNP.

- `pheno.`*suffix* (when simulating under the STD covariate model)

  The covariate data file in BayPass format (see 2.3).

- `poolsize.`*suffix* (when simulating read count data)

  The haploid pool size data file in BayPass format (see 2.3).

4.1.5 Examples

```
#source the baypass R functions (check PATH)
source("utils/baypass_utils.R")
#load the bovine covariance matrix
om.bta <- as.matrix(read.table("examples/omega.bta"))
#simulate allele count data for 1000 SNPs
simu.res <- simulate.baypass(omega.mat=om.bta)
```

```
#simulate allele count data for 1000 neutral SNPs and
#100 associated SNPs with varying regression coefficients
simu.res <- simulate.baypass(omega.mat=om.bta,beta.coef=runif(100,-0.2,0.2),
                             pop.trait=rnorm(18))

#simulate read count data for 1000 SNPs
simu.res <- simulate.baypass(omega.mat=om.bta,coverage=50)
```

4.2 The R function *plot.omega()*

4.2.1 Description

This function performs an eigen-decomposition of the scaled covariance matrix of the population allele frequencies ($\mathbf{\Omega}$ in Figure 1) to allow representation in a two-dimension plot. This actually corresponds to a (between population) PCA–like analysis.

4.2.2 Usage

```
plot.omega(omega,PC=c(1,2),pop.names=paste0("Pop",1:nrow(omega)),
          main=expression("SVD of "*Omega),col=rainbow(nrow(omega)),
          pch=16,pos=2)
```

4.2.3 Arguments

- `omega.mat` (always required)

  A positive definite and symmetric matrix of rank npop corresponding to the covariance matrix of population allele frequencies ($\mathbf{\Omega}$ in Figure 1)

- `PC`

  A vector with two elements correspond to the two Principal Components to be plotted (by default the first two PCs are plotted)

- `pop.names`

  A vector of length *npop* with the names of the populations (should be of the same size as the matrix rank)

- `main`

  Title of the plot

- `col`

  The colors for points and text representing populations. Multiple colors can be specified so that each point can be given its own color. If there are fewer colors than points they are recycled in the standard fashion

- `pch`

  Plotting characters or symbols

4.2.4 Values

The function returns a plot and a list containing i) a matrix of the *npop* PC's (matrix named "PC"); ii) a vector with the npops eigenvalues (vector named "eig"); and iii) a vector with the percentage of variance explained by each PC (vector named "pcent.var")

4.2.5 Example

```
#source the baypass R functions (check PATH)
source("utils/baypass_utils.R")
#load the bovine covariance matrix
om.bta <- as.matrix(read.table("examples/omega.bta"))
pops=c("AUB","TAR","MON","GAS","BLO","MAN","MAR","LMS","ABO",
"VOS","CHA","PRP","HOL","JER","NOR","BRU","SAL","BPN")
om.bta.svd=plot.omega(omega=om.bta,pop.names=pops)
```

4.3 The R function *fmd.dist()*

4.3.1 Description

This function computes the metric proposed by Förstner and Moonen (2003) to evaluate the distance between two covariance matrices (FMD distance).

4.3.2 Usage

```
fmd.dist(mat1,mat2)
```

4.3.3 Arguments

- `mat1` and `mat2`

  Two positive-definite (symmetric) matrices

4.3.4 Values

The function returns a numeric value corresponding to the FMD distance between the two matrices.

4.3.5 Example

```
#source the baypass R functions (check PATH)
source("utils/baypass_utils.R")
#load the bovine covariance matrix
om.bta <- as.matrix(read.table("examples/omega.bta"))
#create a dummy diagonal covariance matrix
#this might be obtained from a star-shaped phylogeny with
#branch length (Fst) equal to 0.1
star.bta<-diag(0.1,nrow(om.bta))

#compute the fmd.dist between the two matrices
fmd.dist(om.bta,star.bta)
```

4.4 The R function *geno2YN()*

4.4.1 Description

This function reads the allele (or read) count data file in the BayPass format and extract both the counts for the reference allele and total counts.

4.4.2 Usage

```
geno2YN(genofile)
```

4.4.3 Arguments

- `genofile`

  A character string giving the name of the allele (or read) count data file in the BayPass format

4.4.4 Values

The function produces a list containing the two following matrices:

- `YY`

  A matrix with *nsnp* rows and *npop* columns containing allele (or read) counts for the reference allele.

- `NN`

  A matrix with *nsnp* rows and *npop* columns containing the total allele (or read) counts.

4.4.5 Example

```
#source the baypass R functions (check PATH)
source("utils/baypass_utils.R")
#load the bovine BTA 14 data
counts.obj <- geno2YN("examples/geno.bta14")
```

4.5 The R function *simulate.PCcorrelated.covariate()*

4.5.1 Description

This function simulates covariate values for each population that are correlated to a given Principal Component of the matrix $\mathbf{\Omega}$ (see Frachon *et al.*, 2018, for an application).

4.5.2 Usage

```
simulate.PCcorrelated.covariate(omega,axis=1,targeted.rho=0.1,tol=0.01)
```

4.5.3 Arguments

- `omega` (always required)

  A positive definite and symmetric matrix of rank *npop* corresponding to the covariance matrix of population allele frequencies ($\mathbf{\Omega}$ in Figure 1)

- `axis`

  The Principal Component number which the simulated covariate should be correlated with

- `targeted.rho`

  The (targeted) Pearson correlation coefficient between the PC and the simulated covariate

- `tol`

The accepted tolerance for the realized Pearson correlation coefficient between the PC and the simulated covariate. Simulations are performed until $\max(-1, \rho-\tau) < \hat{\rho} < \min(1, \rho+\tau)$ where ρ is the targeted correlation (defined by the argument `targeted.rho`); τ is the tolerance (defined by the argument `tau`); and $\hat{\rho}$ is the realized Pearson correlation coefficient between the PC and the covariate.

4.5.4 Values

This function returns a vector of length *npop* containing the simulated covariate values.

4.5.5 Example

```
#source the baypass R functions (check PATH)
source("utils/baypass_utils.R")
#load the bovine covariance matrix
om.bta <- as.matrix(read.table("examples/omega.bta"))
sim.cov0.5<-simulate.PCcorrelated.covariate(omega=om.bta,targeted.rho=0.5)
```

5 Worked Examples

For illustration purposes, in the following different types of analyses based on the example files included in the `example` directory (see 2.3) are detailed step by step.

5.1 Cattle allele count data

5.1.1 Analysis under the core model mode

The following command allows analyzing the data under the core model (this should take from 3 to 4 min on a standard computer):

```
baypass -gfile geno.bta14 -outprefix anacore
```

To visualize the results, one may open an R session and proceed as follows:

```
#source the baypass R functions (check PATH)
source("utils/baypass_utils.R")

#upload the estimated Omega matrix
omega=as.matrix(read.table("anacore_mat_omega.out"))
pop.names=c("AUB","TAR","MON","GAS","BLO","MAN","MAR","LMS","ABO",
            "VOS","CHA","PRP","HOL","JER","NOR","BRU","SAL","BPN")
dimnames(omega)=list(pop.names,pop.names)

# Visualization of the matrix
```

```
    # Using SVD decomposition
    plot.omega(omega=omega,pop.names=pop.names)

    # as a correlation plot
    require(corrplot)
    cor.mat=cov2cor(omega)
    corrplot(cor.mat,method="color",mar=c(2,1,2,2)+0.1,
          main=expression("Correlation map based on"~hat(Omega)))

    # as a heatmap and hierarchical clustering tree (using the average agglomeration method)
    hclust.ave <- function(x) hclust(x, method="average")
    heatmap(1-cor.mat,hclustfun = hclust.ave,
      main=expression("Heatmap of "~hat(Omega)~"("*d[ij]*"=1-"*rho[ij]*")"))

#Compare the estimates of Omega obtained with whole genome data
#and with the BTA14 SNPs only (included in the example file)

wg.omega <- as.matrix(read.table("examples/omega.bta")) #check the PATH
plot(wg.omega,omega) ; abline(a=0,b=1)
fmd.dist(wg.omega,omega)

#Estimates of the XtX differentiation measures (using the calibrated XtXst estimator)
anacore.snp.res=read.table("anacore_summary_pi_xtx.out",h=T)
#check behavior of the p-values associated to the XtXst estimator
hist(10**(-1*anacore.snp.res$log10.1.pval.),freq=F,breaks=50)
abline(h=1)
layout(matrix(1:2,2,1))
plot(anacore.snp.res$XtXst)
plot(anacore.snp.res$log10.1.pval.,ylab="XtX P-value (-log10 scale)")
abline(h=3,lty=2) #0.001 p--value theshold
```

If the p–values are not well behaved[12], one may rather consider calibrating the XtX statistics with PODs (see 5.3). In addition, it should be noticed that for the XtX statistics, the p-values are computed assuming a bilateral test (see 3.3). Hence, one may check the XtX value to distinguish positive (high XtX) from balancing (small XtX) selection.

5.1.2 Analysis under the IS covariate mode *(MCMC is run under the core model)*

In this example, an association analysis with the SMS Morphological Score available for the 18 cattle breeds (see Gautier, 2015) is carried out under the STD covariate model by estimating SNP-specific Bayes Factor and empirical Bayesian P-value (and the underlying regression coefficient) using an Importance Sampling algorithm (see 3.1.2):

```
baypass -gfile geno.bta14 -efile bta.pc1 -outprefix anacovis
```

In other words, the parameters of interest are sampled by running the core model as above (5.1.1). Hence, providing the same seed and the same options

[12]e.g., see the following URL: http://varianceexplained.org/statistics/interpreting-pvalue-histogram/

were used, the same estimates for $\mathbf{\Omega}$ (e.g., files `anacovis_mat_omega.out` and `anacore_mat_omega.out`) and other parameters in common are obtained than under the previous analysis (5.1.1). If covariables are available, one may then consider this mode as the default mode.

Continuing the above example in R, one may plot the Importance Sampling estimates of the Bayes Factor, the empirical Bayesian P-value and the underlying regression coefficient as follows:

```
covis.snp.res=read.table("anacovis_summary_betai_reg.out",h=T)
graphics.off()
layout(matrix(1:3,3,1))
plot(covis.snp.res$BF.dB.,xlab="SNP",ylab="BFis (in dB)")
plot(covis.snp.res$eBPis,xlab="SNP",ylab="eBPis")
plot(covis.snp.res$Beta_is,xlab="SNP",ylab=expression(beta~"coefficient"))
```

Recall that in the example, only a subset of SNPs mapping to BTA14 are considered. To improve precision, one may rather provide the program with the more accurate estimate of $\mathbf{\Omega}$ relying on the complete data set (with 40 times as many SNPs):

```
baypass -gfile geno.bta14 -efile bta.pc1 \
        -omegafile omega.bta -outprefix anacovis2
```

The resulting Importance Sampling estimates of the Bayes Factor, the empirical Bayesian P-value and the underlying regression coefficient might be plotted as follows:

```
covis2.snp.res=read.table("anacovis2_summary_betai_reg.out",h=T)
graphics.off()
layout(matrix(1:3,3,1))
plot(covis2.snp.res$BF.dB.,xlab="SNP",ylab="BFis (in dB)")
plot(covis2.snp.res$eBPis,xlab="SNP",ylab="eBPis")
plot(covis2.snp.res$Beta_is,xlab="SNP",ylab=expression(beta~"coefficient"))
```

5.1.3 Analysis under the MCMC covariate mode *(MCMC is run under the STD model)*

In this example, the association study with the breed SMS Morphological Score is carried out under the STD covariate model to estimate the empirical Bayesian P-value and the underlying regression coefficient (Gautier, 2015). Although one may also estimate $\mathbf{\Omega}$ under the STD model, this option has been inactivated in BayPass. As a consequence, an estimate of $\mathbf{\Omega}$ (e.g., as obtained by a first analysis under the core model or IS covariate mode) must be provided.

```
baypass -gfile geno.bta14 -efile bta.pc1 \
        -covmcmc -omegafile omega.bta -outprefix anacovmcmc
```

The resulting estimates of the empirical Bayesian P-values, the underlying regression coefficients (posterior mean) and the XtX (corrected for the coveriable effect) might be plotted as follows:

```
covmcmc.snp.res=read.table("anacovmcmc_summary_betai.out",h=T)
covmcmc.snp.xtx=read.table("anacovmcmc_summary_pi_xtx.out",h=T)$M_XtX
graphics.off()
layout(matrix(1:3,3,1))
plot(covmcmc.snp.res$eBPmc,xlab="SNP",ylab="eBPmc")
plot(covmcmc.snp.res$M_Beta,xlab="SNP",ylab=expression(beta~"coefficient"))
plot(covmcmc.snp.xtx,xlab="SNP",ylab="XtX corrected for SMS")
```

5.1.4 Analysis under the AUX covariate mode: MCMC is run under the AUX model

In this example, the association study with the breed SMS Morphological Score is carried out under the AUX covariate model to estimate the Bayes Factor (and the underlying regression coefficient). Although one may also estimate $\mathbf{\Omega}$ under the AUX model, this option has been inactivated in BayPass. As a consequence, an estimate of $\mathbf{\Omega}$ (e.g., as obtained by a first analysis under the core model or the IS covariate mode) must be provided.

```
baypass -gfile geno.bta14 -efile bta.pc1 \
        -auxmodel -omegafile omega.bta -outprefix anacovaux
```

The resulting estimates of the Bayes Factor, the underlying regression coefficients (posterior mean) and the corrected XtX might be plotted as follows:

```
covaux.snp.res=read.table("anacovaux_summary_betai.out",h=T)
covaux.snp.xtx=read.table("anacovaux_summary_pi_xtx.out",h=T)$M_XtX
graphics.off()
layout(matrix(1:3,3,1))
plot(covaux.snp.res$BF.dB.,xlab="SNP",ylab="BFmc (in dB)")
plot(covaux.snp.res$M_Beta,xlab="SNP",ylab=expression(beta~"coefficient"))
plot(covaux.snp.xtx,xlab="SNP",ylab="XtX corrected for SMS")
```

To refine the association signal, one may further introduce spatial dependency among SNPs by setting $\mathrm{b_{is}} = 1$ in the Ising prior (Figure 1C) :

```
baypass -gfile geno.bta14 -efile bta.pc1 -auxmodel \
        -isingbeta 1.0 -omegafile omega.bta -outprefix anacovauxisb1
```

The resulting estimates of the Posterior Inclusion Probability (i.e., the posterior mean of the auxiliary variable δ_i) under the AUX models without and with SNP spatial dependency may be plotted as follows:

```
covauxisb1.snp.res=read.table("anacovauxisb1_summary_betai.out",h=T)
graphics.off()
layout(matrix(1:2,2,1))
plot(covaux.snp.res$PIP,xlab="SNP",ylab=expression(delta[i]),main="AUX model")
plot(covauxisb1.snp.res$PIP,xlab="SNP",ylab=expression(delta[i]),
     main="AUX model with isb=1")
```

When including SNP spatial dependency in the model (i.e., $b_{is} > 0$), it may be worth filtering the data set for SNPs displaying low polymorphism across the population (as evaluated by the parameter π in Figure 1). Indeed, nearly fixed SNPs (e.g., $\pi < 0.05$ or $\pi > 0.95$) are not expected to be associated with any covariable even if they are neighboring strongly associated SNPs.

5.2 Littorina Pool–Seq read count data

5.2.1 Analysis under the IS covariate mode

The Littorina Pool–Seq data set may be analyzed in a similar fashion as the cattle data set above except that one needs to specify the (haploid pool) size file using the `-poolsizefile` option to activate the Pool–Seq mode. Because the haploid pool sizes are relatively large ($n = 100$), one may also increase the initial δ of the y_{ij} proposal distribution (as a rule of thumbs, one may set it to a fifth of the minimum pool size). Here is an example of a command to run BayPass under the IS covariate mode (MCMC run under the core model):

```
baypass -gfile lsa.geno -efile lsa.ecotype \
        -poolsizefile lsa.poolsize -d0yij 20 -outprefix lsacovis
```

5.2.2 Contrast Analysis to identify SNPs associated with population ecotypes

The population ecotype being a binary trait (either "crab" or "wave"), one may rely on the C_2 statistic (Olazcuaga *et al.*, 2019) to identify SNPs associated with the population ecotype rather than relying on the (parametric) models used to estimate Bayes Factor. Here is an example of a command to run BayPass to estimate both the C_2 contrast statistic and the BFis[13]:

```
baypass -gfile lsa.geno -poolsizefile lsa.poolsize -d0yij 20 \
        -contrastfile lsa.ecotype -efile lsa.ecotype         \
        -outprefix lsacontrast
```

[13]The estimations of the C_2 and BFis may be done separately but their joint estimation adds almost no extra computational cost and is strictly equivalent. Indeed, in both cases the model parameters are sampled under the core model

The resulting C_2 contrasts (and BF) might then be plotted (and compared) as follows:

```
lsa.ecotype.bf=read.table("lsacontrast_summary_betai_reg.out",h=T)$BF.dB.
lsa.ecotype.C2=read.table("lsacontrast_summary_contrast.out",h=T)
#check the behavior of the p-values associated to the C2
hist(10**(-1*lsa.ecotype.C2$log10.1.pval.),freq=F,breaks=50)
abline(h=1)
plot(lsa.ecotype.bf,lsa.ecotype.C2$log10.1.pval.,
       xlab="BF",ylab="C2 p-value (-log10 scale)")
abline(h=3,lty=2)  #0.001 p--value theshold
abline(v=20,lty=2) #BF threshold for decisive evidence (according to Jeffreys' rule)
```

5.3 Calibrating statistics with the simulation and analysis of PODs (pseudo-observed data sets)

As described in Gautier (2015) and mentioned above, pseudo-observed data sets (PODs) might be considered to calibrate the XtX or C_2 estimates most particularly if their derived p-values are not well behaved[12] (and/or the number of analyzed SNPs is small).

For instance, to produce a (small) POD sample with 1,000 SNPs (continuing the cattle example in 5.1.1) we may rely on the `simulate.baypass()` function (see 4):

```
#get estimates (post. mean) of both the a_pi and b_pi parameters of
#the Pi Beta distribution
pi.beta.coef=read.table("anacore_summary_beta_params.out",h=T)$Mean
#upload the original data to obtain total allele count
bta14.data<-geno2YN("geno.bta14")
#Create the POD
simu.bta<-simulate.baypass(omega.mat=omega,nsnp=1000,sample.size=bta14.data$NN,
beta.pi=pi.beta.coef,pi.maf=0,suffix="btapods")
```

Then, one may analyze the newly created POD (data file named "G.btapods" in the example) giving another prefix for the output files:

```
baypass -gfile G.btapods -outprefix anapod
```

Continuing the above example in R, calibration of the XtX and visualization of the results might be done as follows:

```
#########################################################
#Sanity Check: Compare POD and original data estimates
#########################################################
#get estimate of omega from the POD analysis
pod.omega=as.matrix(read.table("anapod_mat_omega.out"))
plot(pod.omega,omega) ; abline(a=0,b=1)
fmd.dist(pod.omega,omega)

#get estimates (post. mean) of both the a_pi and b_pi parameters of
#the Pi Beta distribution from the POD analysis
```

```
pod.pi.beta.coef=read.table("anapod_summary_beta_params.out",h=T)$Mean
plot(pod.pi.beta.coef,pi.beta.coef)  ; abline(a=0,b=1)

#######################################################
#XtX calibration
#######################################################
#get the pod XtX
pod.xtx=read.table("anapod_summary_pi_xtx.out",h=T)$M_XtX
#compute the 1% threshold
pod.thresh=quantile(pod.xtx,probs=0.99)
#add the thresh to the actual XtX plot
plot(anacore.snp.res$M_XtX)
abline(h=pod.thresh,lty=2)
```

Similarly, when considering analysis of association with population-specific covariable under the core model, one may also calibrate of the different measures (BFis, regression coefficients, correlation coefficients, etc.) by analyzing a POD together with the covariables, i.e., with the previous cattle example (5.1.2):

```
baypass -gfile G.pods -efile bta.pc1 -omegafile omega.bta \
        -outprefix podcovis
```

More generally, the PODs distribution may also be used to compute empirical P–values and to derive from them q-values to control for multiple testing (see the `qvalue` package, Storey and Tibshirani, 2003).

6 Some general advices

6.1 Checking convergence by running several independant runs

As for any MCMC analysis, it is recommended to run several independent MCMC (e.g., from 3 to 5), using different seeds for the random number generators (see `-seed` option in 3.2). Comparing the estimates of parameters like $\mathbf{\Omega}$ and statistics like XtX or BF across runs allows ensuring (empirically) that the chains properly converged. For large enough data sets, estimations are generally reproducible for most parameters and statistics. Yet, for measures like the BFis that are based on an Importance Sampling approximation (see 3.1.1), single run estimations may be unstable (in particular when the number of populations is small), it is then recommended to use as an estimate the median computed over several different independent runs (for a real life example see Gautier *et al.*, 2018).

6.2 To sample or not to sample the regression coefficients in association analysis (i.e., BFis or BFmc)?

For association analyses, the advantages of sampling the regression coefficients (i.e., using the STD or AUX models) rather than relying on the Importance Sampling (IS) approximation are discussed in Gautier (2015). Yet, the IS approximation is more computationally efficient, since only parameter samples drawn from the core model are required (i.e., the regression coefficients are not sampled) allowing estimation of Bayes Factor (BFis) at almost no extra computational costs when running the core model (which is needed to estimate the $\mathbf{\Omega}$ matrix). As a consequence also, jointly analyzing several covariables is strictly equivalent to carrying out separate analyses for each covariable in turn. In other words, the Bayes Factors (BFis) estimated for each covariable are associated to a single-covariate regression model while in the case of the STD or AUX models (that involve the sampling of the regression coefficients) the estimated Bayes Factors (BFmc) would correspond to a multiple-covariate regression model if the covariate file include several covariables. Finally, from a practical point of view, using the STD or AUX models with data sets containing a small number of populations (e.g., < 8) is not recommended since some identifiability issues may arise.

6.3 Dealing with large data sets

When dealing with very large data sets ($> 10^6$ SNPs), one may adopt a sub-sampling strategy that consists in analyzing pseudo-"independant" sub-data sets of ca. 50,000 to 100,000 SNPs (e.g., Frachon *et al.*, 2018; Gautier *et al.*, 2018). With *nsnp* SNPs, these sub-data sets can be generated for instance by sampling one every k SNPs in an ordered map then leading to k sub-data sets of ca. $\frac{Nsnps}{k}$ SNPs (the first sub-data set containing SNP numbers $1, k+1, 2k+1, \ldots$; the second sub-data sets containing SNP numbers $2, k+2, 2k+2, \ldots$; and so on). The function `pooldata.subset()` available from the R package `poolfstat` (Hivert *et al.*, 2018) may be used to easily generate such sub-data sets. The k sub-data sets may then be analyzed separately (or in parallel on a computer grid). After comparing the different estimated $\mathbf{\Omega}$ matrices[14] and ensuring that they are similar[15], the various SNP specific statistics may be combined. Such a sub-sampling approach has several advantages.

From a computational point of view, BayPass is not scaling linearly with the number of threads (e.g., Table 1) and above 8 threads, the gain in in-

[14]e.g., using the FMD distance (see 4) or by directly comparing the matrix elements

[15]This is generally the case unless the number of SNPs per sub-sample is too small

creasing the number of threads is almost nil. It is thus (far) more efficient to analyze the k sub-data sets each on a single thread rather than the whole data sets in k threads. From a more general point of view, the sub-sampling approach allows comparing results across (pseudo-independant) sub-data sets, each having, in addition, a lower level of background LD (if a thinning approach has been performed to generate the sub-data sets).

7 Credits

BayPass makes use of several functions and subroutines that were previously developed by other authors. These include:

- the Fortran code for the multiple streams `MT19937` Mersenne–Twister (parallel) Random Number Generator was adapted from the subroutines available in the `mt_stream_f90-1.11.tar.gz` program written by Ken-Ichi Ishikawa and available under the New BSD License[16] at http://theo.phys.sci.hiroshima-u.ac.jp/~ishikawa/PRNG/mt_stream_f90-1.11/).

- Various functions and subroutines for random number generations that were adapted from the Alan Miller Fortran module `random.f90` available at: http://jblevins.org/mirror/amiller/ available under the GNU GPL license.

- the Wishart sampler utilities derived from the fortran `wishart` library written by John Burkhardt and available at http://people.sc.fsu.edu/~%20jburkardt/f_src/wishart/wishart.html under the GNU GPL license.

- the `kracken(3f)` Fortran module developped by John S. Urban to parse command line arguments (available at http://home.comcast.net/~urbanjost/LIBRARY/libCLI/arguments/krackenhelp.html) under the GNU GPL license.

I also wish to thank Renaud Vitalis for providing the LaTeXtemplate for this manual and Andrew Beckerman for reporting bugs and advice that helped improving the program.

[16] http://theo.phys.sci.hiroshima-u.ac.jp/~ishikawa/PRNG/mt_stream_f90-1.11/LICENSE

8 Copyright

9 Contact

If you have any question, please feel free to contact me. However, I strongly recommend you read carefully this manual first.

Bibliography

Coop, G., D. Witonsky, A. D. Rienzo, and J. K. Pritchard, 2010 Using environmental correlations to identify loci underlying local adaptation. Genetics 185: 1411–1423.

Duforet-Frebourg, N., E. Bazin, and M. G. B. Blum, 2014 Genome scans for detecting footprints of local adaptation using a bayesian factor model. Mol Biol Evol 31: 2483–2495.

Frachon, L., C. Bartoli, S. Carrère, O. Bouchez, A. Chaubet, *et al.*, 2018 A genomic map of climate adaptation in arabidopsis thaliana at a micro-geographic scale. Front Plant Sci 9: 967.

Förstner, W., and B. Moonen, 2003 A metric for covariance matrices. In *Geodesy-The Challenge of the 3rd Millennium*. Springer Berlin Heidelberg, 299–309.

Gautier, M., 2015 Genome-wide scan for adaptive divergence and association with population-specific covariates. Genetics 201: 1555–1579.

Gautier, M., J. Yamaguchi, J. Foucaud, A. Loiseau, A. Ausset, *et al.*, 2018 The genomic basis of color pattern polymorphism in the harlequin ladybird. Curr. Biol. 28: 3296–3302.

Genz, A., F. Bretz, T. Miwa, X. Mi, F. Leisch, *et al.*, 2015 *mvtnorm: Multivariate Normal and t Distributions*. R package version 1.0-3.

Günther, T., and G. Coop, 2013 Robust identification of local adaptation from allele frequencies. Genetics 195: 205–220.

Hasselman, B., 2015 *geigen: Calculate Generalized Eigenvalues of a Matrix Pair*. R package 1.5.

Hivert, V., R. Leblois, E. J. Petit, M. Gautier, and R. Vitalis, 2018 Measuring genetic differentiation from pool-seq data. Genetics 210: 315–330.

Nicholson, G., A. V. Smith, F. Jonsson, O. Gustafsson, K. Stefansson, *et al.*, 2002 Assessing population differentiation and isolation from single-nucleotide polymorphism data. J Roy Stat Soc B 64: 695–715.

Olazcuaga, L., A. Loiseau, H. Parinello, M. Paris, A. Fraimout, *et al.*, 2019 A whole-genome scan for association with invasion success in the fruit fly drosophila suzukii using contrasts of allele frequencies corrected for population structure. bioRxiv : https://doi.org/10.1101/851303.

Paradis, E., J. Claude, and K. Strimmer, 2004 Ape: Analyses of phylogenetics and evolution in r language. Bioinformatics 20: 289–290.

Pickrell, J. K., and J. K. Pritchard, 2012 Inference of population splits and mixtures from genome-wide allele frequency data. PLoS Genet 8: e1002967.

R Core Team, 2015 *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Spiegelhalter, D. J., N. G. Best, B. P. Carlin, and A. van der Linde, 2002 Bayesian measures of model complexity and fit. Journal of the Royal Statistical Society. Series B (Statistical Methodology) 64: 583–639.

Storey, J. D., and R. Tibshirani, 2003 Statistical significance for genomewide studies. Proc. Natl. Acad. Sci. U.S.A. 100: 9440–5.

Wei, T., 2013 *corrplot: Visualization of a correlation matrix*. R package version 0.73.

Westram, A. M., J. Galindo, M. A. Rosenblad, J. W. Grahame, M. Panova, *et al.*, 2014 Do the same genes underlie parallel phenotypic divergence in different littorina saxatilis populations? Mol Ecol 23: 4603–4616.