SNPSetSimulations: simulation of genotypic profiles under dependence

Florian Hébert, Mathieu Emily, David Causeur

1 Introduction

This vignette provides help and examples of usage of the functions of the R package `SNPSetSimulations`. This package depends on packages **lattice** [3], **Matrix** [2] and, most importantly, **GenOrd** [1].

In GWAS simulation studies, SNP-sets (or genes for simplicity) are often considered. These correspond to genotype data matrices; such a matrix can be denoted $\boldsymbol{X}$, its generic term being denoted x_{ij}. $x_{ij} \in \{0, 1, 2\}$ is the number of copies of the minor allele for the i-th individual and j-th SNP of the set. Strong and heterogeneous dependence structures are often observed among adjacent SNPs. Simulation studies must replicate such dependence structures to obtain realistic and reliable results.

The **SNPSetSimulations** package aims at giving tools for generating simulated data related to case-control genome-wide association studies. In particular, it includes tools for simulating genotype and phenotype data under realistic scenarios. The package provides four functions: `levelplotSNP`, `PopulationSNPSet`, `PopulationPhenotype`, and `SampleSNPPhenotype`. Usage of each function will be explained hereafter, with examples given for each one.

2 Representing Within-Gene Dependence Structures

The `levelplotSNP` function can be used to represent within-gene dependence structures as an heatmap. This function is based on the `levelplot` function from the **lattice** package [3]. Dependence is measured by Pearson's correlation coefficient. In the following example, we create a correlation matrix `Sigma` of size 20. `Sigma` has an autocorrelation structure with correlation parameter `rho` equal to 0.8. We finally plot this correlation matrix; see figure 1 for the result.

```
m = 20
rho = 0.8
Sigma = rho^abs(outer(1:m,1:m,"-"))
levelplotSNP(Sigma)
```

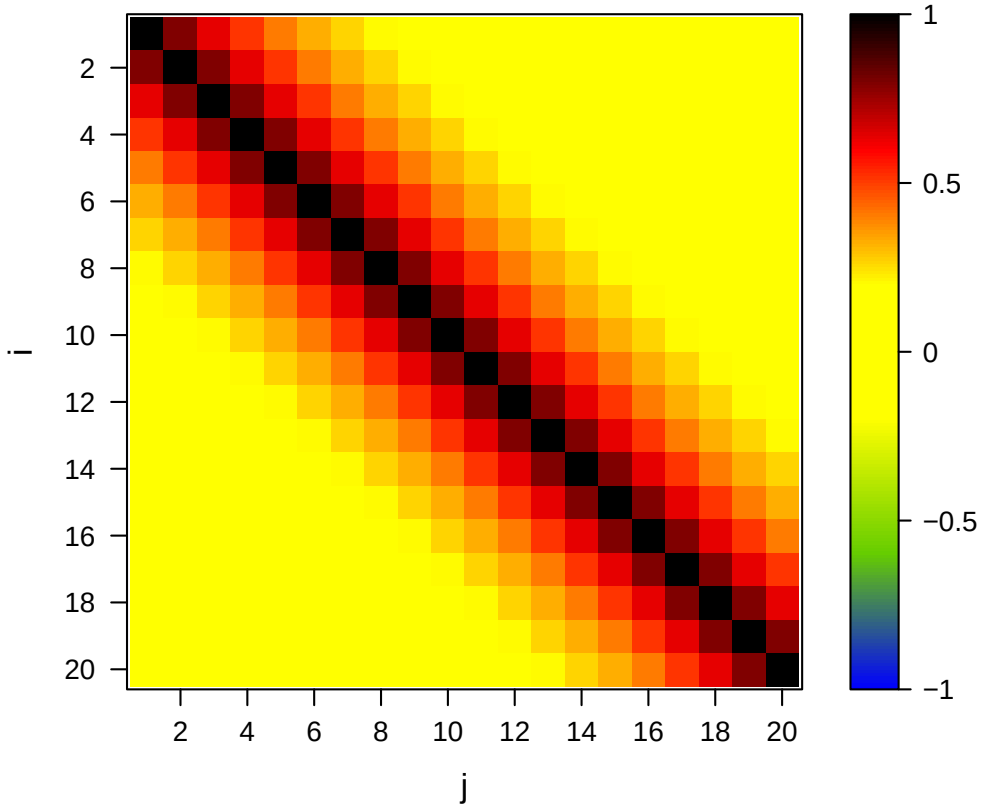

Figure 1: Heatmap of a correlation matrix obtained with function `levelplotSNP`

This function can be useful to display the dependence structure of a genotype matrix `X`. If such a matrix is loaded in the R environment, then it is sufficient to enter the following command:

```
levelplotSNP(cor(X))
```

A legend is also displayed on the right of the plot. The colors are automatically chosen by the function on a continuous scale passing through yellow for null correlation, black for strong positive correlation, red for intermediate positive correlation, blue for strong negative correlation and green for intermediate negative correlation.

3 Generating Genotype Data

The function `PopulationSNPSet` provides a way of efficiently generating a matrix of genotypes for a population of `n` individuals and `m` SNPs according to given marginal distributions

and within-gene dependence structure. This function can be used through several ways explicited by the following examples. In each example, the obtained matrix is stored in the variable `G`. Several arguments can be used:

- `n`: the desired size of the simulated population
- `Sigma`: the correlation matrix of the simulated SNP-set
- `p`: a vector of minor allele frequencies (one value for each SNP); if this argument is given, genotypes are simulated according to Hardy-Weinberg equilibrium (HWE)
- `marginal`: a list of marginal distributions. This argument can be used to generate genotypes under a chosen distribution, which can be different from HWE. Each element of the list is a vector of cumulative marginal probabilities. If the k-th SNP takes values 0, 1 and 2 with probabilities `p0`, `p1` and `p2`, the k-th vector of the list is `c(p0,p0+p1)`. The third cumulative probability is supposed to be equal to one and is not given
- `X`: an observed SNP-set; this argument can be used to generate a population of `n` individuals with the same dependence structure and marginal distributions as those observed in the given SNP-set.

Simulation of `n = 100000` individuals, `m = 10` autocorrelated SNPs (correlation coefficient equal to 0.8), minor allele frequency `p = 0.4` for each SNP, under HWE:

```
G = PopulationSNPSet(n=100000,Sigma=0.8^abs(outer(1:10,1:10,"-")),
    p=rep(0.4,10))
```

Simulation of `n = 100000` individuals, and `m = 10` autocorrelated SNPs (correlation coefficient equal to 0.8). Here, contrary to the first example, where only the MAF of each SNP was given, the complete marginal distribution of each SNP is given. Each SNP is supposed to be uniformly distributed (*i.e.* the probability for an individual to get 0, 1 or 2 copies of the minor allele equals 1/3). These marginal distributions are given in the argument `marginal`, as a list of `m` elements. Each element of the list is a vector of cumulative probabilities.

```
M = cbind(rep(1/3,10),rep(1/3+1/3,10))
M = lapply(1:nrow(M),function(i){M[i,]})
G = PopulationSNPSet(n=100000,Sigma=0.8^abs(outer(1:10,1:10,"-")),
    marginal=M)
```

Simulation of `n = 100000` individuals, and `m` SNPs according to marginal distributions and dependence structure given by an observed SNP set `Xobs`:

```
G = PopulationSNPSet(n=100000,X=Xobs)
```

In the last example, the dependence structure, dimension (number of SNPs) and marginal distributions are automatically estimated on `Xobs`. One can check that the simulated data has the requested dependence structure and marginal distributions by comparing them to the observed SNP-set.

It is not trivial to generate correlated SNP data. Function `PopulationSNPSet` is based on modified versions of functions from the **GenOrd** package [1], which gives tools for generating discrete data with given dependence structure and marginal distributions.

4 Generating Phenotypes

Once genotypes for a population are generated, a disease status corresponding to these genotypes can be generated using a specified model. The `PopulationPhenotype` function generates disease status conditionally to a matrix of genotypes `X` using a logistic model. Let $\boldsymbol{x_i}$ and $\boldsymbol{u_i}$ be the vector of genotypes and the optional vector of covariates of the i-th individual, respectively. Then the i-th individual is diseased with probability π_i with

$$\pi_i = \frac{\exp(\beta_0 + \boldsymbol{u_i'}\boldsymbol{\alpha} + \boldsymbol{x_i'}\boldsymbol{\beta})}{1 + \exp(\beta_0 + \boldsymbol{u_i'}\boldsymbol{\alpha} + \boldsymbol{x_i'}\boldsymbol{\beta})}.$$

The disease status is then sampled as a Bernoulli variable with probability of success π_i. Cases are coded 1 and controls 0. Several parameters must be given:

- `X`: a matrix of genotype data, such as obtained with function `PopulationSNPSet`
- `beta0`: the intercept of the logistic model
- `beta`: the values of the non-zero coordinates of the vector $\boldsymbol{\beta}$ of the logistic model
- `I`: the positions of the non-zero coordinates of $\boldsymbol{\beta}$
- `U`: an optional matrix of covariates
- `alpha`: the vector of effect parameters corresponding to the covariates
- `mod`: a vector of characters giving the association model of each involved SNP - `"A"` for additive association (default), `"R"` for recessive association and `"D"` for dominant association.

In the following example, $\beta_0 = -3$ and the 2nd and 7th coordinates of $\boldsymbol{\beta}$ are non-zero (both equal 0.2). The association between the disease and the involved SNPs corresponds to a recessive model. No covariates are used.

```
Y = PopulationPhenotype(X=G,beta0=-3,beta=c(0.2,0.2),I=c(2,7),mod=c("R","R"))
```

The association models can be mixed (*e.g.* a SNP can be associated to the disease through an additive model and the other one through a recessive model). One can also desire to generate a phenotype that is independent from the SNPs. This corresponds to $\boldsymbol{\beta} = \mathbf{0}$; therefore, it can be done by setting `beta = 0` and `I = 1`. This function is not really supposed to be used on its own, but is necessary for the next function.

5 Generating Samples of Genotype and Phenotype Data

The most useful feature of this package is to generate samples of genotype data and corresponding disease status, *i.e.* samples similar to that used in case-control GWAS. This is what provides the `SampleSNPPhenotype` function. Using a matrix of genotype data corresponding to a population (and an optional matrix of covariates), it generates the corresponding disease status using the `PopulationPhenotype` function. Then, it constructs a sample by randomly sampling a specified number of cases and controls. The result is given as a list of three elements: the first one is the genotype matrix of the sample, the second one is the vector of disease status corresponding to each row of the genotype matrix and the third one is the matrix of covariates corresponding to the sample. Several parameters must be given:

- `X`: a matrix of genotype data
- `beta0`: the intercept of the logistic model
- `beta`: the values of the non-zero coordinates of the vector $\boldsymbol{\beta}$ of the logistic model
- `I`: the positions of the non-zero coordinates of $\boldsymbol{\beta}$
- `n0`: number of controls
- `n1`: number of cases
- `U`: an optional matrix of covariates
- `alpha`: the vector of effect parameters corresponding to the covariates
- `mod`: a vector of characters giving the association model of each involved SNP - `"A"` for additive association (default), `"R"` for recessive association and `"D"` for dominant association.

In the following example, we generate a sample of 1,000 cases and 1,000 controls with the same genotype data and the same association model as in the example for function `PopulationPhenotype`:

```
A = SampleSNPPhenotype(X=G,beta0=-3,beta=c(0.2,0.2),I=c(2,7),n0=1000,
    n1=1000,mod=c("R","R"))
```

The elements of the list are named `SNP`, `Phenotype` and `Covariates`. One can then compute association tests between each SNP and the phenotype. For example, using the χ^2 test:

```
Z = apply(A$SNP,2,function(x){chisq.test(x,A$Phenotype,correct=FALSE)$p.value})
```

References

[1] Barbiero, A. and Ferrari, P. A. (2015). *GenOrd: Simulation of Discrete Random Variables with Given Correlation Matrix and Marginal Distributions*. R package version 1.4.0.

[2] Bates, D. and Maechler, M. (2018). *Matrix: Sparse and Dense Matrix Classes and Methods*. R package version 1.2-14.

[3] Sarkar, D. (2008). *Lattice: Multivariate Data Visualization with R*. Springer, New York. ISBN 978-0-387-75968-5.