Package Ravages (RAre Variant Analysis and GEnetic Simulation), Simulations

Herve Perdry and Ozvan Bocher

2020-10-09

Introduction

Ravages was developped to simulate genetic data and to perform rare variant association tests (burden tests and the variance-component test SKAT) on more than two groups of individuals (Bocher et al., 2019, Genetic Epidemiology). Ravages relies on the package Gaston developped by Herve Perdry and Claire Dandine-Roulland. Most functions are written in C++ thanks to the packages Rcpp, RcppParallel and RcppEigen.
Functions of Ravages use bed.matrix to manipulate genetic data as in the package Gaston (see documentation of this package for more details).
In this vignette, we illustrate how to perform genetic simulations and how to use it to compute power of rare variant association tests. See the main vignette for more details about rare variant association tests.
To learn more about all options of the functions, the reader is advised to look at the manual pages.
We develpped two main simulations procedures, one based on allelic frequencies and genetic relative risks (GRR), and the other one based on haplotypes and a liability model. All the functions can be used to simulate more than two groups of individuals.

Simulations based on allelic frequencies and GRR

Calculation of frequencies in each group of individuals

The first step to simulate genetic data is to compute genotypic frequencies in each group of individuals based on frequencies in the general population and on genetic relative risk (GRR) values. GRR correspond to the increased risk of a disease for a given genotype compared to a reference genotype, here the homozygous genotype for the reference allele. More precisely, the GRR associated to the heterozygous genotype in the group c can be calculated as follow:

$$GRR_{Aa} = \frac{P(Y = c|Aa)}{P(Y = c|AA)}$$

With Y the phenotype (1 for the controls, and c going from 2 to C in the cases), A the reference allele, and a the alternate allele. The frequency of each genotype in each group of cases c can be calculated using Bayes theorem:

$$P(Aa|Y = c) = \frac{P(Y = c|Aa)P(Aa)}{\sum_{Geno=AA,Aa,aa} P(Y = c|Geno)P(Geno)} = \frac{GRR_{Aa} \times P(Aa)}{P(AA) + GRR_{Aa} \times P(Aa) + GRR_{aa} \times P(aa)}$$

P(AA), P(Aa) and P(aa) corresponding to the genotypic probabilities in the general population. The three genotypic frequencies can then be calculated in the controls group using the rule of total probability:

$$P(Geno|Y = 1) = P(Geno) - \sum_{c=2}^{c=C} P(Geno|Y = c)P(Y = c)$$

The function **genotypic.freq()** performs these calculations to obtain the three genotypic frequencies in the different groups of individuals. To do so, the user needs to give *P(Y=c)*, the prevalence of each group of

cases (argument *prev*), and the GRR values. GRR values need to be in a matrix form with one row per cases group and one column per variant. If there is no supposed link between the GRR associated to the heterozygous genotype and the GRR associated to the homozygous alternative genotype (general model of the disease, *genetic.model* = "*general*"), the user needs to specify two GRR matrices: one for GRR_{Aa} (argument *GRR.het*) and one for GRR_{aa} (argument *GRR.homo.alt*). If *genetic.model=“recessive”*, *“multiplicative”* or *“dominant”*, only one GRR matrix is needed. **genotypic.freq()** will return a list with three matrices, one for each genotype containing the genotypic frequencies, with one row per group of individuals and one column per variant.
To help the user with the construction of the GRR matrix for **genotypic.freq()**, we implemented the function **GRR.matrix()**.
To use this function, the user needs to specify how the GRR should be calculated (argument *GRR*):

- the same GRR is given to all the variants (*GRR=“constant”*), its value being specified to *GRR.value*;
- the GRR is computed using the same formula as in SKAT: $w = -0.4 * |log_{10}(MAF)|$ (*GRR=“SKAT”*);
- the GRR is computed using another function depending on MAF in the general population (*GRR=“variable”*), this function being specified to *GRR.function*.

In the two last situations, a file containing the MAF in the general population with at least a column “maf” and a column “gene” should be given to the argument *genes.maf*. Two such files are available in Ravages: the file *Kryukov* containing MAF simulated under the demographic model of Kryukov and the file *GnomADgenes* containing MAF from the NFE population in GnomAD. As these files contain MAF for multiple genes, the user needs to specify which gene to choose to simulate the data with the argument *select.gene*. If this argument is empty, only the first gene will be kept in the simulation procedure.
Finally, the multiplicative factor of the GRR between each group of cases compared to the first group of cases needs to be specified to the argument *GRR.multiplicative.factor* (number of values: number of cases groups - 1).
GRR.matrix() will return a GRR matrix in the appropriate format for the function **genotypic.freq()**.
Examples of these two functions are shown below:

```
#GRR calculated using the same formula as in SKAT,
#with values in the second group of cases being twice the values from the first one

GRR.del <- GRR.matrix(GRR = "SKAT", genes.maf = Kryukov, n.case.groups = 2,
                      GRR.multiplicative.factor=2, select.gene = "R1")
GRR.del[,1:5]
```

```
##           [,1]      [,2]     [,3]      [,4]      [,5]
## [1,]  5.037728  6.222656 12.34472  9.042877  8.133663
## [2,] 10.075455 12.445313 24.68944 18.085755 16.267327
```

```
#Calculation of genotype frequencies in the two groups of cases and the controls group
#The previous GRR matrix is used with a multiplicative model of the disease
#The prevalence in each group of cases is 0.001

geno.freq.groups <- genotypic.freq(genes.maf = Kryukov, select.gene="R1",
                                   GRR.het = GRR.del, prev = c(0.001, 0.001),
                                   genetic.model = "multiplicative")
str(geno.freq.groups)
```

```
## List of 3
##  $ freq.homo.ref: num [1:3, 1:383] 1 0.999 0.998 1 1 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "controls" "cases_1" "cases_2"
##   .. ..$ : NULL
##  $ freq.het     : num [1:3, 1:383] 1.87e-04 9.56e-04 1.91e-03 5.57e-05 3.52e-04 ...
```

```
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "controls" "cases_1" "cases_2"
##   .. ..$ : NULL
##  $ freq.homo.alt: num [1:3, 1:383] 7.90e-09 2.29e-07 9.15e-07 6.49e-10 3.11e-08 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "controls" "cases_1" "cases_2"
##   .. ..$ : NULL
```

```
#frequencies of the homozygous alternative genotype in the different groups
geno.freq.groups$freq.homo.alt[,1:5]
```

```
##                  [,1]         [,2]         [,3]         [,4]         [,5]
## controls 7.897334e-09 6.485888e-10 7.480447e-14 6.568600e-12 2.503543e-11
## cases_1  2.288672e-07 3.106821e-08 4.778956e-11 9.067311e-10 2.469545e-09
## cases_2  9.145933e-07 1.242291e-07 1.911556e-10 3.626706e-09 9.877199e-09
```

It is also possible to calculate the MAF in each group of individuals as follow:

```
#MAF calculation for the first five variants
geno.freq.groups$freq.homo.alt[,1:5] + 0.5*geno.freq.groups$freq.het[,1:5]
```

```
##                  [,1]         [,2]         [,3]         [,4]         [,5]
## controls 9.375276e-05 2.785699e-05 5.403418e-07 3.246158e-06 5.972867e-06
## cases_1  4.784006e-04 1.762618e-04 6.912999e-06 3.011198e-05 4.969452e-05
## cases_2  9.563437e-04 3.524614e-04 1.382590e-05 6.022214e-05 9.938410e-05
```

Simulation of genotypes

In addition to compute the genotypic frequencies in each group of individuals, it is possible to directly simulate genotypes for a group of controls and more than two groups of cases. This can be done using the function **random.bed.matrix()** which relies on the function **genotypic.freq()** explained previously. The arguments *genes.maf*, *select.gene*, *prev* and *genetic.model* are the same as in **genotypic.freq()**.
In **random.bed.matrix()**, the proportion of causal variants and protective variants (among causal variants) simulated in the genomic region should be specified to *p.causal* and *p.protect* respectively. The argument *GRR.matrix.del* should contain a matrix with GRR values as if all the variants were deleterious. If *genetic.model="general"*, two GRR matrices need to be given as a list to the argument *GRR.matrix.del* (one for the heterozygous genotype and the other for the homozygous alternative genotype).
If the user wants to simulate protective variants in addition to deleterious variants, a similar argument to *GRR.matrix* should be given to *GRR.matrix.pro* with GRR values as if all variants were protective. If the argument *GRR.matrix.pro* is empty and *p.protect*>0, *GRR.matrix.pro* will correspond to $1/GRR.matrix$.
These deleterious and protective GRR values will then be assigned to the sampled deleterious and protective variants in the simulations, the non-causal variants having GRR values of 1.
The size of the different groups of individuals should be a vector specified to *size*, and the user has to choose whether the causal variants will be the same between the different groups of cases with the argument *same.variant*.
Finally, the number of simulations need to be specified to *replicates*.
random.bed.matrix() will return a bed matrix with the group of each individual in the field *@ped$pheno*, the first one being considered by default as the controls group, and the replicate number corresponding to the genomic region in the field *@snps$genomic.region*.
The example below shows how to simulate a group of 1 000 controls and two groups of 500 cases with 50% of deleterious variants having GRR values from the previous example. The deleterious variants are different between the two groups of cases and 5 genomic regions are simulated.

```
x <- random.bed.matrix(genes.maf = Kryukov, size = c(1000, 500, 500), replicates = 5,
                       prev = c(0.001, 0.001), GRR.matrix.del = GRR.del,
                       p.causal = 0.5, p.protect = 0, same.variant = FALSE,
```

```
                        genetic.model = "multiplicative", select.gene = "R1")
x
```

```
## A bed.matrix with 2000 individuals and 1915 markers.
## snps stats are set
##   There are  1648  monomorphic SNPs
## ped stats are set
```

```
table(x@ped$pheno)
```

```
##
##    0    1    2
## 1000  500  500
```

```
table(x@snps$genomic.region)
```

```
##
##  R1  R2  R3  R4  R5
## 383 383 383 383 383
```

Simulations based on haplotypes

Ravages also offers the possibility to perform genetic simulations based on real haplotypes data to mimic linkage desequlibrium pattern observed in these data.
Two options are available to do so: genetic data are simulated using either haplotypes and their frequencies in different groups of individuals (function **rbm.haplos.freqs()**), or using haplotypes and a liability model on their burdens (function **rbm.haplos.thresholds()**).
For the first situation, **rbm.haplos.freqs()** requires a matrix of haplotypes (argument *haplos* with one row per haplotype and one column per variant), and a matrix with each haplotype frequency in each simulated group of individuals (argument *freqs* with one row per haplotype and one column per group of individuals). Observed haplotypes are then sampled in each group of individuals in respect to their frequencies in each one of them.
For the second situation, **rbm.haplos.thresholds()** requires a matrix of haplotypes (argument *haplos* as well), and a number of other arguments to draw the liability model. The idea of these simulations is to draw a liability distribution of the burden of causal variants for each haplotype, to compute the probability of each haplotype in each group of individuals based on this distribution, and finally to sample haplotypes in each group based on those probabilities. *weights* is a vector whose length corresponds to the number of variants and is used to weight variant into burdens computation (by default, the same formula as in SKAT is used, which gives the higher weights to the rarest variants : $w = -0.4 * |log_{10}(MAF)|$). *maf.threshold* corresponds to the maf threshold used to define a rare variants, i.e. all monomorphic variants and variants with a MAF upper this threshold will have a weight of 0. In addition, the number of causal variants needs to be given to *nb.causal*, and the proportion of protective variants among causal variants needs to be given to *p.protect*. Causal variants will be sampled among variants with a positive weight, and haplotypes' burden will be computed on these variants.
h2 corresponds to the phenotypic variance explained by the gene: haplotypes burdens will be adjusted on this value; and *prev* corresponds to the prevalence of each group of individuals and will be used to compute the liability threshold for each group of individuals (individuals in group i will fall between 1-$prev_i$ and +Inf). This quantile is drawn from a standard normal distribution if *normal.approx*=TRUE (applicable for low *h2* value and pretty large *prev* values), otherwise, quantiles will be drawn from a distribution based on 1e6 observed sampled burdens. If one wants to simulate a group of unselected controls, *prev* needs to be set at 1, regardless of the other arguments: haplotypes are sampled uniformly in the whole liability distribution. *nb.causal*, *p.protect*, *h2* and *prev* should be vector of equal size as the number of groups. If they are of size 1, their value will be repeated.
As for the previous functions, the number of simulations to preform should be given to *replicates*. As

haplotypes' probabilies can take some time to run for a set of causal variants, it is possible to run several multiple replicates on this same set. Indeed, even if the same causal variants are taken, different haplotypes (with same probabilities) will be sampled in each individual for each replicate. To ensure a minimal variability, we yet recommend to resample the set of causal variants several times. For example, for 1000 replicates, we recommend to resample causal variants 20 times, and therefore to perform 50 replicates by set of causal variants. The number of replicates to perform by set of causal variants should be given to *rep.by.causal*.

As for the genetic simulations based on GRR with **rbm.haplos.freqs()** or **rbm.haplos.thresholds()**, the sizes of each group should be given to *size*, and the number of simulations to *replicates*. As before, the replicate number will be in the slot *@snps$genomic.region* and the group of individuals in *@ped$pheno*, both being factors. If **rbm.haplos.freqs()** is used, phenotype values will correspond to the colnames of *freqs*.

Two examples to illustrate these functions are presented below. The first one enables the simulation of 5 groups of individuals, with haplotype frequencies from the five european populations for the LCT gene. The second one enables the simulation of one group of 100 controls and two groups of 50 cases based on the liability model with different genetic effects. Parameters for these simulations are obtained using **get.rbm.thresholds()**

```
#Load LCT dataset for haplotype matrix
data(LCT.haplotypes)

#Simulation based on haplotypes frequencies
#for the variants in the LCT gene in the EUR population
LCT.gene.hap <- LCT.hap[which(LCT.sample$super.population=="EUR"),
                        which(LCT.snps$pos>=136545410 & LCT.snps$pos<=136594750)]
#Individuals from EUR
LCT.sample.EUR <- subset(LCT.sample, super.population=="EUR")
#Matrix of haplotype frequencies
LCT.freqs <- sapply(unique(LCT.sample.EUR$population), function(z)
                    ifelse(LCT.sample.EUR$population==z,
                           1/table(LCT.sample.EUR$population)[z], 0))
#Simulation of genetic data for five groups of 50 individuals
x <- rbm.haplos.freqs(haplos=LCT.gene.hap, freqs=LCT.freqs, size=rep(50,5), replicates=5)

#Simulation of 100 controls, and two groups of 50 cases with 30 causal variants
#and with the second group having half h2 and twice the prevalence
#compared to the first one
#5 replicates are performed and causal variants are sampled once
x <- rbm.haplos.thresholds(haplos=LCT.gene.hap, nb.causal = 30, h2=c(0.01,0.01,0.02),
                           prev=c(1,0.01,0.005), size=c(100, 50, 50), replicates=5,
                           rep.by.causal = 5)
```

```
## 3  groups of individuals will be simulated
```

Power calculation

Power calculations are not directly implemented into **Ravages**, but the functions *burden()* and *SKAT()* can be used to perform burden tests and SKAT. To have more informations about these two functions, please refer to the main vignette of the package. After the simulation of a bedmatrix using one of the two previously explained methods, statistical power can be computed as follows:

```
#Simulations using GRR values
#The two groups of cases are genetically heterogeneous
GRR.del <- GRR.matrix(GRR = "SKAT", genes.maf = Kryukov, n.case.groups = 2,
                      GRR.multiplicative.factor=2, select.gene = "R1")
```

```
x.GRR <- random.bed.matrix(genes.maf = Kryukov, size = c(200, 100, 100),
                           prev = c(0.001, 0.001), GRR.matrix.del = GRR.del,
                           p.causal = 0.3, p.protect = 0, same.variant = TRUE,
                           replicates = 100, genetic.model = "multiplicative",
                           select.gene = "R1")
table(x.GRR@ped$pheno)
```

```
##
##   0   1   2
## 200 100 100
```

```
#Selection of rare variants: MAF<1% in the entire sample
x.GRR.filter <- filter.rare.variants(x.GRR, maf.threshold=0.01, filter="whole")

#Comparing each group of cases to the controls with WSS
x.GRR.1 <- select.inds(x.GRR.filter, x.GRR.filter@ped$pheno %in% c(0,1))
x.GRR.2 <- select.inds(x.GRR.filter, x.GRR.filter@ped$pheno %in% c(0,2))
#Construction of null models
H0.cas1 <- NullObject.parameters(x.GRR.1@ped$pheno, ref.level = 0,
                                 RVAT = "burden", pheno.type = "categorial")
H0.cas2 <- NullObject.parameters(x.GRR.2@ped$pheno, ref.level = 0,
                                 RVAT = "burden", pheno.type = "categorial")
#Run WSS
wss.cas1 <- burden(x.GRR.1, H0.cas1, burden="WSS", cores = 1)
```

```
## Categorial phenotype
```

```
mean(wss.cas1$p.value<0.05)
```

```
## [1] 0.19
```

```
wss.cas2 <- burden(x.GRR.2, H0.cas2, burden="WSS", cores = 1)
```

```
## Categorial phenotype
```

```
mean(wss.cas2$p.value<0.05)
```

```
## [1] 0.53
```

```
#Power of SKAT with each group considered separately
#Null model
skat.null.3gps <- NullObject.parameters(x.GRR.filter@ped$pheno,
                                        RVAT = "SKAT", pheno.type = "categorial")
skat.3gps <- SKAT(x.GRR.filter, skat.null.3gps,
                  params.sampling = list(perm.target = 100, perm.max = 5e4))
```

```
## Categorial phenotype
## permutations
```

```
mean(skat.3gps$p.value<0.05)
```

```
## [1] 0.27
```