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

Herve Perdry and Ozvan Bocher

2019-09-17

```
library("knitr")
require("Ravages")
```

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 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:

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

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 alternate 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 fas 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 population NFE 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 twice the once 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 alternate homozygous 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 five first 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 alternate 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 protective and deleterious GRR values will then be assigned to the sampled protective and deleterious 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 should choose whether the causal variants will be the same between the different groups of cases with the argument *same.variant*.
Finally, the number of genomic regions simulated is specified with the argument *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  1636  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 haplotypes to mimic linkage desequlibrium pattern observed on these data.

This is performed using the function **simulated.bedmatrix.haplo()** which is based on a matrix of observed haplotypes with one row per haplotype and one column per variant (argument *haplos*). The number (argument *nb.causal*) or the proportion (arguments *p.causal*) of causal variants are sampled from observed variants for each simulation replicate. It is also possible to add protective variants with the argument *p.protect* which corresponds to the proportion of protective variants among causal variants. The burden of each haplotype is then computed by weighting the causal variants according to a function given to the argument *weights*. By default, the same formula as in SKAT is used, which gives the higher weights to the rarest variants : $w = -0.4 * |log_{10}(MAF)|$.
Once these burdens are calculated, they are adjusted on a given *h2* value defined by the user which represents the variance of the phenotype explained by the gene. Burdens are considered under a liability model, and threshold from the standard gaussian distribution are considered to represent the prevalence of each sub-phenotype (argument *prev*). These thresholds are then used to compute the probability of each haplotype in each group of individuals, and pairs of haplotypes are finally sampled for each individual according to these previous conditionnal probabilities.
In addition to these arguments, **simulated.bedmatrix.haplo()** needs other information: as for the simulations based on the GRRs, the sizes of the different groups should be given to the argument *size*, and the number of replicates to the argument *replicates*. The arguments *p.causal*/*nb.causal*, *p.protect*, *h2*, *prev* and *size* should have the same length as the total number of groups.
To simulate a group of controls, *prev* needs to be set as 1 in this group of individuals, regardless of the other arguments. By modifying the different arguments, the user can simulate genetic data under various scenarios: simulate similar genetic effects between the groups of cases (same *h2* and *prev*), simulate one group of cases as the controls group (*prev*=1 in this group of cases), …
As before, the phenotype of each individual can be found in the field *@ped$pheno*, and the replicate number can be found in the field *@snps$genomic.region*.

Below is an exemple of data simulation of one group of controls and two groups of cases, with genetic heterogeneity between the two groups of cases:

```
#Load LCT dataset for haplotype matrix
#Selection of LCT gene (positions from GnomAD) and EUR superpop
data(LCT.haplotypes)
```

```
id.LCT <- subset(LCT.snps, pos>=136545410 & pos <= 136594750)$id
haplo.matrix <- LCT.hap[which(LCT.sample$super.population=="EUR"),id.LCT]

#Simulation of 200 controls, 100 individuals from two sub-phenotypes
#with the second grave having half h2 and twice the prevalence compared to the first one
#30 causal variants are present
x <- simulated.bedmatrix.haplo(haplos=haplo.matrix, nb.causal=30, h2=c(0.01, 0.01, 0.02),
                               prev=c(1, 0.01, 0.005), size=c(200,100,100), replicates=5)
x
```

```
## A bed.matrix with 400 individuals and 7145 markers.
## snps stats are set
##   There are  5712  monomorphic SNPs
## ped stats are set
```

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

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

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

```
## 
##   R1   R2   R3   R4   R5 
## 1429 1429 1429 1429 1429
```

Power calculation

Power calculations are not directly implemented into **Ravages**, but the functions *burden.mlogit()* and *SKAT()* can be used to compute the power of 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 previous methods explained, you can compute statistical power as follows:

```
#Simulations using GRR values
#The three groups of cases are genetically homogeneous
GRR.del <- GRR.matrix(GRR = "SKAT", genes.maf = Kryukov, n.case.groups = 3,
                      GRR.multiplicative.factor=c(1,1), select.gene = "R1")

x.GRR <- random.bed.matrix(genes.maf = Kryukov, size = c(200, 100, 100, 100),
                          prev = c(0.001, 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   3 
## 200 100 100 100
```

```
#Comparing each group of cases to the controls
x.GRR.1 <- select.inds(x.GRR, x.GRR@ped$pheno %in% c(0,1))
x.GRR.2 <- select.inds(x.GRR, x.GRR@ped$pheno %in% c(0,2))
x.GRR.3 <- select.inds(x.GRR, x.GRR@ped$pheno %in% c(0,3))
skat.cas1 <- SKAT(x.GRR.1, maf.threshold = 1)
skat.cas2 <- SKAT(x.GRR.2, maf.threshold = 1)
skat.cas3 <- SKAT(x.GRR.3, maf.threshold = 1)
mean(skat.cas1<0.05) ; mean(skat.cas2<0.05) ; mean(skat.cas3<0.05)
```

```
## [1] 0.2925
```

```
## [1] 0.195
```

```
## [1] 0.235
```

```
#Simulations based on haplotypes
#One group of controls and two groups of cases genetically heterogeneous
x <- simulated.bedmatrix.haplo(haplos=haplo.matrix, nb.causal=2, h2=c(0.01, 0.025, 0.05),
                               prev=c(1, 0.025, 5e-3), size=c(200,100,100), replicates=100)
#Power of the burden test CAST using the multinomial regression: each group is
#considered separately; significance level=5%
cast <- burden.mlogit(x, burden="CAST", ref.level=0)$p.value
mean(cast<0.05)
```

```
## [1] 0.21
```

```
#Considering all the cases as one group and compare them to the controls
pheno.pooled <- ifelse(x@ped$pheno == 0, 0, 1)
cast.pooled <- burden.mlogit(x, group=pheno.pooled, burden="CAST", ref.level=0)$p.value
mean(cast.pooled<0.05)
```

```
## [1] 0.05
```

```
#Power of the burden test WSS comparing each group of cases to the controls
#at a 1% significance threshold
x.1 <- select.inds(x, x@ped$pheno != 2)
x.2 <- select.inds(x, x@ped$pheno != 1)
wss.cas1 <- burden.mlogit(x.1, burden="WSS", ref.level=0)$p.value
wss.cas2 <- burden.mlogit(x.2, burden="WSS", ref.level=0)$p.value
mean(wss.cas1<0.01) ; mean(wss.cas2<0.01)
```

```
## [1] 0.01
```

```
## [1] 0.47
```