Package Ravages (RAre Variant Analysis and GEnetic Simulation)

Herve Perdry and Ozvan Bocher

2019-07-10

```
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 rare variant association tests on real data. A second vignette is available showing how to simulate genetic data and calculate power of the tests. To learn more about all options of the functions, the reader is advised to look at the manual pages.

Example of analysis using LCT data

Below is an example of an association analysis and previous steps of data filtering using the dataset LCT available with the package Gaston. This dataset containts data from the 1000Genome project in the region containing the LCT gene. In this example, we look for an association between rare variants and the populations of 1000Genomes. The population of each individual is available in LCT.pop. Details about each function is given right after this example.

```
#Importation of data in a bed matrix
x <- as.bed.matrix(LCT.gen, LCT.fam, LCT.bim)

# Group variants within known genes
x <- set.genomic.region(x)
table(x@snps$genomic.region, useNA = "ifany")
```

```
##
## R3HDM1  UBXN4    LCT   MCM6   DARS   <NA>
##    165     91    128     67     49    107
```

```
# Group variants within know genes by extending their positions
# 500bp upstream and downstream
x <- set.genomic.region(x, flank.width=500)
table(x@snps$genomic.region, useNA = "ifany")
```

```
##
## R3HDM1  UBXN4    LCT   MCM6   DARS   <NA>
##    165     93    130     69     49    101
```

```
# Filter variants with maf (computed on whole sample) < 0.025
# keeping only genomic region with at least 10 SNPs
x1 <- filter.rare.variants(x, filter = "whole", maf.threshold = 0.025, min.nb.snps = 10)
table(x1@snps$genomic.region, useNA="ifany")
```

```
##
## R3HDM1  UBXN4    LCT
##     19     10     19
```

```
# run burden test WSS, using the 1000Genome population as "outcome"
burden.mlogit(x1, group=LCT.pop, burden = "WSS", ref.level = "CEU")
```

```
##             p.value is.err
## R3HDM1 2.020269e-05      0
## UBXN4  6.440681e-05      0
## LCT    2.393502e-09      0
```

```
# run SKAT, using the 1000Genome population as "outcome"
SKAT(x1, group=LCT.pop)
```

```
##            stat      p.perm       p.chi2      p.value
## R3HDM1 2.785458 1.99996e-05 4.848424e-08 4.848424e-08
## UBXN4  1.968380 1.99996e-05 1.646702e-05 1.646702e-05
## LCT    3.336874 1.99996e-05 5.083867e-09 5.083867e-09
```

Defining genomic regions

For rare variant association tests, the unit of analysis is not a single variant but a genomic region, typically a gene. The first step of the analysis is therefore to group variants, which can be done using gene positions with the function **set.genomic.region()**. It works on a bed.matrix (see Gaston) and simply adds a column "genomic.region" to the slot x@snps containing the gene assigned to each variant. By default, any variant being outside the gene positions won't be annotated. Gene positions can be extended to annotate more variants using the argument *flank.width* corresponding to the number of base pair upstream and downstream the gene. If *flank.width=Inf*, each variant will be assigned to the nearest gene. If two genes overlap, variants in the overlapping zone will be attributed to the second one (in the order given by the position of their starting point on the genome).
The files **genes.b37** and **genes.b38** available in Ravages which contain gene positions from ENSEMBL versions hg19 and hg38 can be used to define gene positions.

Rare variant definition

To perform rare variant analysis, it is also important to define what is a rare variant in order to leave out common ones. The function **filter.rare.variants()** enables to keep only variants of interest based on a given MAF threshold while leaving out monomorphic variants. This function uses and returns a bed.matrix which can be filtered in three different ways:

- If *filter="whole"*, all the variants with a MAF lower than the threshold in the entire sample will be kept.
- If *filter="controls"*, all the variants with a MAF lower than the threshold in the controls group will be kept. In this situation, the controls group needs to be specified to the argument *ref.level*.
- If *filter="any"*, all the variants with a MAF lower than the threshold in any of the groups will be kept.

It is also possible to specify the minimum number of variants needed in a genomic region to keep it using the parameter *min.nb.snps*.

Rare variant association tests

We have implemented two rare variant burden association tests extensions: CAST and WSS. The general idea of burden tests is to compute a genetic score per individual and per genomic region and and to test if it differs between the different groups of individuals. To extend these tests to more than two groups of individuals, a non-ordinal multinomial regression is used. The independant variable in this regression is the genetic effect of the gene represented by the genetic score and potential covariates can be added in the model. In addition to the genetic scores CAST and WSS directly implemented in the package, the user can specify another genetic score for the regression.

Genetic score for burden tests

We have implemented two functions to compute CAST and WSS scores respectively. These functions return a matrix with one row per individual and one column by genomic region. They are directly called in the function **burden.mlogit()** if these scores are used to perform the association tests.

CAST

CAST is based on a binary score which has a value of one if an individual carries at least one variant in the considered genomic region, and 0 otherwise. A MAF threshold for the definition of a rare variant is therefore needed as an argument to *maf.threshold*. This score can be computed using the function **CAST()** as shown here on the LCT data:

```r
#Calculation of the genetic score with a maf threshold of 1%
CAST.score <- CAST(x = x1, genomic.region = x1@snps$genomic.region, maf.threshold = 0.025)
head(CAST.score)
```

```
##         R3HDM1 UBXN4 LCT
## HG00096      0     0   0
## HG00097      1     0   0
## HG00099      0     0   0
## HG00100      1     0   0
## HG00101      1     0   1
## HG00102      0     0   0
```

WSS

WSS (Weighted Sum Statistic) is based on a continuous score giving the highest weights to the rarest variants as follow:

$$WSS_j = \sum_{i=1}^{R} I_{ij} * w_i$$

with

$$w_i = \frac{1}{\sqrt{(t_i * q_i * 1 - qi)}}$$

and

$$q_i = \frac{n_i + 1}{2 * t_i + 1}$$

Where n_i is the total number of minor alleles genotyped for variant i, t_i is the total number of alleles genotyped for variant i and I_{ij} is the number of minor alleles of variant i for the invidual j. In the original method, each variant is weighted according to its frequency in the controls group. In our version of WSS, the weights depend on allele frequency calculated on the entire sample. The function **WSS()** can be used to compute the WSS score as shown on the LCT data:

```
WSS.score <- WSS(x = x1, genomic.region = x1@snps$genomic.region)
head(WSS.score)
```

```
##            R3HDM1 UBXN4       LCT
## HG00096 0.0000000     0 0.0000000
## HG00097 0.2812769     0 0.0000000
## HG00099 0.0000000     0 0.0000000
## HG00100 1.2060026     0 0.0000000
## HG00101 0.2812769     0 0.6734209
## HG00102 0.0000000     0 0.0000000
```

Regressions

We have extended CAST and WSS using non-ordinal multinomial regression models. Let consider C groups of individuals including a group of controls ($c = 1$) and $C - 1$ groups of cases with different sub-phenotypes of the disease. We can compute $C - 1$ probability ratios, one for each group of cases:

$$ln\frac{P(Y_j = c)}{P(Y_j = 1)} = \beta_{0,c} + \beta_{G,c}X_G + \beta_{k1,c}K_1 + ... + \beta_{kl,c}K_l$$

Where Y_j corresponds to the phenotype of the individual j and K_l is a vector for the lth covariate with the corresponding coefficient β_{kl}. The genetic effect is represented by X_G and correspond to the genetic score CAST or WSS with $\beta_{G,c}$ the log-odds ratio associated to this burden score.
The p-value associated to the genetic effect is calculated using a likelihood ratio test comparing this model to the same model without the genetic effect (null hypothesis). If only two groups are compared, a classical logistic regression is performed.
This regression can be performed on a bed.matrix using the function **burden.mlogit()** which relies on the package mlogit. To do so, the user needs to specify a vector with the phenotype of each individual (argument *group*) and the gene of each variant (argument *genomic.region*).
The CAST or WSS genetic scores can be directly calculated in the regression (*burden="CAST"* or *burden="WSS"*). The user can also use another genetic score in the regression, which has to be specified as a matrix with one individual per row and one genomic region per column to *burden*. In this situation, no bed matrix is needed. The reference group of individuals should be given to the argument *ref.level*. Potential covariates could also be included in the regression as a matrix with one row per individual and one column per covariate to the argument *data*. If only a subset of covariates from *data* are to be included in the model, a R formula should be given to *formula* with these covariates, otherwise all the covariates will be included.
burden.mlogit() will return the p-value associated to the regression for each genomic region. If there is a convergence problem with the regression, the function will return 1 in the column *is.err*. The odds ratio associated to each group of cases compared to the reference group (*ref.level*) with its confidence interval at a given alpha threshold (argument *alpha*) can also be obtained if *get.OR.value=TRUE*.
An example of the p-value and OR calculation with its 95% confidence interval for WSS on the LCT data is shown below using either the argument *burden="WSS"* or using directly the score matrix computed by the function **WSS()**. The outcome here corresponds to the population of the individuals from 1000Genome.

```
#WSS
burden.mlogit(x=x1, group=LCT.pop, burden="WSS", ref.level="CEU", alpha=0.05, get.OR.value=TRUE)
```

```
##             p.value is.err   OR.FIN    OR.GBR   OR.IBS   OR.TSI
## R3HDM1 2.020269e-05      0 3.443069 0.7409557 2.299083 7.122056
## UBXN4  6.440681e-05      0 1.785671 2.4902843 1.534822 6.967061
## LCT    2.393502e-09      0 1.907257 0.8020219 2.935282 8.544081
##        l.lower.FIN l.lower.GBR l.lower.IBS l.lower.TSI l.upper.FIN
## R3HDM1   1.2043461   0.2164537   0.7938208    2.599452    9.843286
## UBXN4    0.3702730   0.5542578   0.3103196    1.757549    8.611535
## LCT      0.6336618   0.2189973   1.0453521    3.201638    5.740651
##         l.upper.GBR l.upper.IBS l.upper.TSI
## R3HDM1     2.536410    6.658662    19.51322
## UBXN4     11.188864    7.591140    27.61796
## LCT        2.937201    8.242086    22.80124
```

```
#Simulation of covariates
#with different probabilities in GBR/CEU/FIN and IBS/TSI
set.seed(1)
covar <- data.frame( sex = c(sample(0:1, sum(table(LCT.pop)[c("CEU", "GBR", "FIN")]), TRUE, c(0.2,0.8))
                             sample(0:1, sum(table(LCT.pop)[c("TSI", "IBS")]), TRUE, c(0.8,0.2))),
                     u = runif(length(LCT.pop)))

#Regression with the covariate "sex"
burden.mlogit(x=x1, group=LCT.pop, burden="WSS", ref.level="CEU", alpha=0.05, get.OR.value=TRUE, data=c
```

```
##             p.value is.err   OR.FIN    OR.GBR   OR.IBS   OR.TSI
## R3HDM1 1.702228e-05      0 2.968668 0.6387658 1.999296 7.101557
## UBXN4  3.953080e-04      0 1.741311 2.4776320 1.484661 6.460990
## LCT    6.166425e-08      0 2.328084 0.9811515 3.560722 8.613469
##         l.lower.FIN l.lower.GBR l.lower.IBS l.lower.TSI l.upper.FIN
## R3HDM1    0.9521956   0.1695159   0.6334386    2.583150    9.255438
## UBXN4     0.3371111   0.5197667   0.2797927    1.652713    8.994552
## LCT       0.7195488   0.2474839   1.1864829    3.186828    7.532462
##         l.upper.GBR l.upper.IBS l.upper.TSI
## R3HDM1     2.406981    6.310295    19.52350
## UBXN4     11.810415    7.878041    25.25810
## LCT        3.889781   10.685990    23.28078
```

```
#WSS using directly the score matrix computed previously
burden.mlogit(burden=WSS.score, group=LCT.pop, ref.level="CEU", alpha=0.05, get.OR.value=TRUE)
```

```
##             p.value is.err   OR.FIN    OR.GBR   OR.IBS   OR.TSI
## R3HDM1 2.020269e-05      0 3.443069 0.7409557 2.299083 7.122056
## UBXN4  6.440681e-05      0 1.785671 2.4902843 1.534822 6.967061
## LCT    2.393502e-09      0 1.907257 0.8020219 2.935282 8.544081
##         l.lower.FIN l.lower.GBR l.lower.IBS l.lower.TSI l.upper.FIN
## R3HDM1    1.2043461   0.2164537   0.7938208    2.599452    9.843286
## UBXN4     0.3702730   0.5542578   0.3103196    1.757549    8.611535
## LCT       0.6336618   0.2189973   1.0453521    3.201638    5.740651
##         l.upper.GBR l.upper.IBS l.upper.TSI
## R3HDM1     2.536410    6.658662    19.51322
## UBXN4     11.188864    7.591140    27.61796
## LCT        2.937201    8.242086    22.80124
```

SKAT

We also extended the variance-component test SKAT using a geometric interpretation. Unlike the burden tests, the is no burden calculated in this test, but the distribution of the genetic effect in the tested gene is compared to a null distribution. SKAT is based on a linear mixed model where the random effects correspond to the genetic effects.
Permutations are used to compute the p-values with the corresponding arguments *perm.target* and *perm.max*. In this function, the covariates can be included using the argument *Pi*. This arguments should be a matrix containing the probabilities that each individual belongs to each group, with one row per individul and one column per group of individuals. It can be directly computed using the function **Pi.matrix()** which has the same arguments *group*, *data*, *formula* and *ref.level* than **burden.mlogit()**. This function uses a regression

with the *group* as the dependant variable and the covariates as the independant variables, and returns the adjusted probabilities. *ref.level* won't have any impact on the probability of each individual, but is needed to perform the regression.
An example of this function and how to use to result in SKAT is shown below.

```
#Compute the Pi matrix
Pi.matrix.LCT <- Pi.matrix(group=LCT.pop, data=covar, formula= ~ sex, ref.level="CEU")

#SKAT with the covariates
SKAT(x1, group=LCT.pop, Pi=Pi.matrix.LCT)
```

```
##              stat       p.perm       p.chi2      p.value
## R3HDM1 2.588570 0.0000199996 7.577283e-06 7.577283e-06
## UBXN4  1.594581 0.0041627169 4.197082e-03 4.162717e-03
## LCT    2.876050 0.0001999960 1.217575e-04 1.217575e-04
```

Data management

Data in the plink format or in the vcf format can be loaded in R using the functions **read.bed.matrix()** and **read.vcf()** respectively from the package gaston.
If the data for the controls and the different groups of cases are in different files, they can be loaded separately and then combined using the function **gaston:::rbind()** as long as the same variants are present between the different groups of individuals.
An example is given below where the simulated data have been split according the the group of each individual, and then combined in a bed.matrix:

```
#Selection of each group of individuals
CEU <- select.inds(x1, LCT.pop=="CEU")
CEU
```

```
## A bed.matrix with 99 individuals and 48 markers.
## snps stats are set
##   There are  6  monomorphic SNPs
## ped stats are set
```

```
FIN <- select.inds(x1, LCT.pop=="FIN")
FIN
```

```
## A bed.matrix with 99 individuals and 48 markers.
## snps stats are set
##   There are  15  monomorphic SNPs
## ped stats are set
```

```
GBR <- select.inds(x1, LCT.pop=="GBR")
GBR
```

```
## A bed.matrix with 91 individuals and 48 markers.
## snps stats are set
##   There are  13  monomorphic SNPs
## ped stats are set
```

```
#Combine in one file:
CEU.FIN.GBR <- rbind(CEU, FIN, GBR)
CEU.FIN.GBR
```

```
## A bed.matrix with 289 individuals and 48 markers.
## snps stats are set
## ped stats are set
```