Gene-Based Methods to Detect Gene-Gene Interaction in R: The GeneGeneInteR Package

Mathieu Emily, Nicolas Sounac, Florian Kroell, Magalie Houée-Bigot

Journal of Statistical Software

October 2020, Volume 95, Issue 12. doi: 10.18637/jss.v095.i12

Gene-Based Methods to Detect Gene-Gene Interaction in R: The GeneGeneInteR Package

Mathieu Emily
Agrocampus Ouest
IRMAR UMR 6625

Nicolas Sounac
Agrocampus Ouest

Florian Kroell
Agrocampus Ouest

Magalie Houée-Bigot
Agrocampus Ouest

Abstract

GeneGeneInteR is an R package dedicated to the detection of an association between a case-control phenotype and the interaction between two sets of biallelic markers (single nucleotide polymorphisms or SNPs) in case-control genome-wide associations studies. The development of statistical procedures for searching gene-gene interaction at the SNP-set level has indeed recently grown in popularity as these methods confer advantage in both statistical power and biological interpretation. However, all these methods have been implemented in home made softwares that are for most of them available only on request to the authors and at best have a web interface. Since the implementation of these methods is not straightforward, there is a need for a user-friendly tool to perform gene-based gene-gene interaction. The purpose of **GeneGeneInteR** is to propose a collection of tools for all the steps involved in gene-based gene-gene interaction testing in case-control association studies. Illustrated by an example of a dataset related to rheumatoid arthritis, this paper details the implementation of the functions available in **GeneGeneInteR** to perform an analysis of a collection of SNP sets. Such an analysis aims at addressing the complete statistical pipeline going from data importation to the visualization of the results through data manipulation and statistical analysis.

Keywords: association studies, gene-gene interaction, dependence, single nucleotide polymophism.

1. Introduction

Case-control genome-wide association studies (GWAS) have become a standard tool for the

identification of susceptibility loci underlying common complex diseases. Single-locus approaches, whereby a large number of single nucleotide polymorphisms (SNPs) are tested independently for association, are usually performed in a first step analysis of GWAS. A large number of softwares have therefore been developed to handle GWAS data and to accomplish whole genome single association testing. When considering that individuals are randomly sampled among healthy and diseased populations, the most popular tools are the independent softwares **PLINK** (Purcell *et al.* 2007) and **SNPStats** (Solé, Guinó, Valls, Iniesta, and Moreno 2006), the R (R Core Team 2020) packages **GenABEL** (Aulchenko, Ripke, Isaacs, and van Duijn 2007), **SNPassoc** (Gonzalez *et al.* 2007), **adegenet** (Jombart and Ahmed 2011), **GWASTools** (Gogarten *et al.* 2012) and **postgwas** (Hiersche, Rühle, and Stoll 2013) as well as R **Bioconductor** (Gentleman *et al.* 2004) packages **snpStats** (Clayton 2020) and **CGEN** (Bhattacharjee, Chatterjee, Han, Song, and Wheeler 2019). Although single-locus approaches have been successfully applied in many studies, findings have explained relatively little of the heritability of most complex traits. The "missing heritability" can be partly explained by several features such as the presence of rare variants, the interaction with the environment, the observation of sex-specific effects, the location of SNPs in non-coding regions, the indirect measure of effects through tag-SNP observation, the presence of common variants with truly small effects. In this article we focus on two complementary sources of missing heritability: (1) SNP-set association and (2) SNPxSNP interaction testing for which efforts have been made in the literature.

In SNP-set association testing, all the SNPs within the region of a gene (or more generally a locus) are jointly modeled as a set. Since the gene is considered as the functional unit of the genome, SNP-set testing has gain in popularity the last few years and has been implemented in most of the above cited softwares (**PLINK**, **SNPStats**, **GenABEL**, **SNPAssoc**, **postgwas**). Additional softwares have been dedicated to the implementation of SNP-sets methods such as **PUMA** (Hoffman, Logsdon, and Mezey 2013) and the R packages **hapassoc** (Burkett, Graham, and McNeney 2006), **cpvSNP** (McHugh, Larson, and Hackney 2020), **SKAT** (Lee, Zhao, Miropolsky, and Wu 2020), **MixMAP** (Matthews and Foulkes 2015) and **aSPU** (Kwak *et al.* 2016).

SNPxSNP interaction consists in testing the association between the case-control status and the interaction between two SNPs. Such testing allow the detection of an epistatic effect between two loci which is very often considered as a main source of missing heritability. Therefore, popular softwares have proposed an implementation of SNPxSNP interaction testing in randomly-sampled design (**PLINK**, **GenABEL**, **SNPStats SNPAssoc** and **CGEN**). Many other tools have also been developed to tackle the issue of detecting SNPxSNP interaction, as, for example, the independent programs **BOOST** (Wan *et al.* 2010) and **TEAM** (Zhang, Huang, Zou, and Wang 2010) as well as the R packages **MDR** (Winham 2012), **logicFS** (Schwender 2020) and **etma** (Lin 2016).

However, none of these softwares and, to our knowledge, no R package has implemented functionality to simultaneously tackle the two objectives of SNP-set association and SNPxSNP interaction into a SNP-set $\times$ SNP-set interaction test. This lack of computational tools to test for interaction at the SNP-set level in randomly-sampled case/control designs is a clear limitation in the quest of missing heritability since gene-level testing can help characterizing functional, compositional and statistical interactions (Phillips 2008). By aggregating signals across variants in a SNP-set with respect to the linkage disequilibrium (LD) structure, statistical power is likely to be increased in situations when multiple causal interactions influence

the phenotype of interest (Huang, Chanda, Alonso, Bader, and Arking 2011; Wu *et al.* 2010). Furthermore, if the interacting variants are only tagged, rather than directly observed, gene-based tests can aggregate signals from different tag SNPs in partial LD. Finally, the use of the gene as the statistical unit can greatly facilitate the biological interpretation of findings (Jorgenson and Witte 2006; Neale and Sham 2004).

However, from a statistical point-of-view, detecting interaction at the SNP-set level is challenging, thus explaining the lack of computational tools devoted to SNP-set × SNP-set interaction. Tackling the two issues of SNP-set association and interaction indeed requires the simultaneous modeling of the correlation within and between the two SNP sets. Nevertheless, the very recent years have seen the development of statistical methods dedicated to the detection of interaction at the SNP-set level.

These methods can be grouped into two main classes. In a first class, proposed methods aim at modeling the joint distribution of SNPs within and between two genes through principal component analysis (*PCA*, Li, Tang, Biernacka, and de Andrade 2009), composite linkage disequilibrium (*CLD*, Rajapakse, Perlman, Martin, Hansen, and Kooperberg 2012), canonical correspondence analysis (*CCA*, Peng, Zhao, and Xue 2010), kernel canonical correspondence analysis (*KCCA*, Larson *et al.* 2014), partial least square path modeling (*PLSPM*, Zhang *et al.* 2013) and gene-based information gain method (*GBIGM*, Li *et al.* 2015). A second class of methods aims at aggregating interaction tests performed at the SNP level by the minimum p value (*minP*, Emily 2016), a gene-based association test using extended Simes (*GATES*, Li, Gui, Kwan, and Sham 2011), and two truncated tests, the truncated tail strength (*tTS*, Jiang, Zhang, Zuo, and Kang 2011) and the truncated product p values (*tProd*, Zaykin, Zhivotovsky, Westfall, and Weir 2002). However, all these methods have been implemented in home made softwares that are for most of them available only on request to the authors and at best have a web interface. Thus, searching for gene-gene interaction at the gene level is not straightforward. Furthermore, a comprehensive comparison of such methods, in terms of power and computational performances, remains hardly feasible.

In this paper, we present the R package **GeneGeneInteR** (Emily, Sounac, Kroell, and Houee-Bigot 2020) that provides an assembly of tools for all the steps involved in gene-based gene-gene interaction testing in case-control association studies when dealing with randomly-sampled individuals. Our package, available from **Bioconductor** (`https://bioconductor.org/packages/GeneGeneInteR/`) and from GitHub (`https://github.com/MathieuEmily/GeneGeneInteR`), proposes a set of functions that allow to (1) download data in various standardized formats (PED, PLINK, VCF), (2) impute missing genotypes, (3) perform a gene-based gene-gene interaction analysis and (4) visualize the results. Section 2 provides an overview of the implementation of the package and aims at summarizing the dependencies to other R packages. In Section 3, technical details and implementation are given regarding the 10 above mentioned statistical procedures (PCA, CLD, CCA, KCCA, PLSPM, GBIGM, minP, GATES, tTS, tProd) proposed in **GeneGeneInteR**. Section 4 is devoted to the description of the functions implemented in **GeneGeneInteR** in order to analyze a collection of SNP sets. Through the example of a dataset related to rheumatoid arthritis and composed of 17 genes, we introduce tools for the importation and the manipulation of the data as well as the visualization of the results.

2. Overview of the implementation of GeneGeneInteR

The implementation of **GeneGeneInteR** can be divided into three main functionality modules. The first module is a set of functions dedicated to the importation and the manipulation of raw data. This module exploits functionalities attached to the class '`SnpMatrix`'. '`SnpMatrix`' is a S4 class of objects developed under the **snpStats** package that is dedicated to large SNP association studies. A detailed example of the functionalities of this module is proposed in Sections 4.1 and 4.2. Our second module consists in functions devoted to the visualization of the results. One functionality of this module allows a network representation, for which functionalities from the **igraph** package (Csardi and Nepusz 2006) are imported with **GeneGeneInteR**. The functionalities of this module are detailed in Section 4.4. Finally, our third module is a set of functions used to perform the statistical analysis. The main function of this module is the function `GGI` that allows the statistical testing of gene-gene interactions in a set of genes.

The implementation of `GGI` is displayed in the flowchart of Figure 1 and can be detailed as follows. First, function `GGI` takes as input three main parameters: the case-control status `Y`, a '`snpMatrix`' object `snpX` obtained with the functions of our first module and a character string `method` that indicates which statistical pairwise method has to be performed. The object `snpX` contains a collection of n genes and the purpose of the `GGI` function is to iteratively test the $n(n-1)/2$ possible pairs of genes with a for loop. For each pair of genes, the pairwise interaction is tested according to the statistical method specified in the `method` argument. The output of `GGI` is an object of class '`GGInetwork`' defined as an S3 class consisting of a list of 4 elements: `statistic`, `p.value`, `method` and `parameter`. Using our own class allows us to define our own functionalities for printing, summarizing and plotting the result of the analysis of a set of genes. To this aim, we define specific S3 methods `print`, `summary` and `plot` that are adapted to the type of statistical analysis performed by the function `GGI`. A detailed example of the use of `GGI` is provided in Section 4.3.

In **GeneGeneInteR**, a total of 10 pairwise methods are available corresponding to statistical

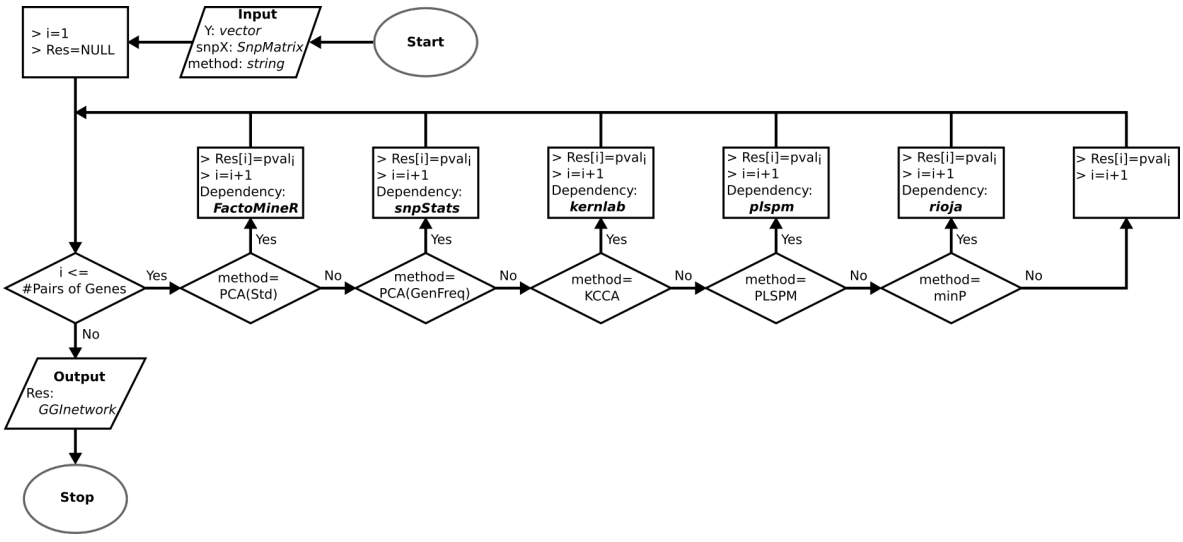

Figure 1: Flowchart of the implementation of the main function `GGI`.

procedures introduced in the literature. Most of these methods are based on general statistical methods that have to be adapted to (1) the specific context of SNP-set interaction testing and (2) the type of data representation used in **GeneGeneInteR**. Therefore, the 10 pairwise methods have been implemented in separate functions that have their own dependency to other R packages. For example, if `method = "PCA"`, the pairwise function `PCA.test` utilizes functionalities from packages **FactoMineR** (Lê, Josse, and Husson 2008) and **snpStats**. As usual, the specific numeric results obtained might differ for these multivariate methods depending on the linear algebra library used. For ease of use and convenience, the mandatory input arguments as well as the output argument are identical for each pairwise function: The input contains three mandatory arguments (`Y`, `G1` and `G2`) and the output is an object of class '`htest`'. A formal statistical description as well as examples of usage are provided for each pairwise method in Section 3.

3. Statistical methods for testing one pair of genes

In this paper, we propose an R package that implements statistical procedures to test for the interaction between two genes in susceptibility with a binary phenotype, typically a case/control disease status. Let $Y \in \{0, 1\}$ be the phenotype, where $Y = 0$ stands for a control and $Y = 1$ a case, and X_1 and X_2 be the two genes for which the interaction is tested. Let consider a sample of n individuals with n_c controls and n_d cases ($n_c + n_d = n$) and $\mathbf{Y} = [y_1, \ldots, y_n]^\top$ the vector of the observed binary phenotypes. Individuals are assumed to be randomly sampled among the two populations of cases and controls. Each gene is a collection of respectively m_1 and m_2 SNPs. The observed genotypes for gene X_1 can be represented by a $n \times m_1$ matrix: $\mathbf{X}_1 = [x_{ij}^1]_{i \in \{1,\ldots,n\}; j \in \{1,\ldots,m_1\}}$ where $x_{ij}^1 \in \{0; 1; 2\}$ is the number of copies of the minor allele for SNP j carried by individual i. A similar representation is used for gene X_2 where $\mathbf{X}_2$ is a $n \times m_2$ matrix. Let us further introduce $\mathbf{X}_1^c$ and $\mathbf{X}_2^c$ the matrices of observed genotypes among controls for gene X_1 and X_2 and $\mathbf{X}_1^d$ and $\mathbf{X}_2^d$ among cases for both genes. Thus $\mathbf{X}_1^c$ is a $n_c \times m_1$ matrix, $\mathbf{X}_1^d$ a $n_d \times m_1$ matrix, $\mathbf{X}_2^c$ a $n_c \times m_2$ matrix and $\mathbf{X}_2^d$ a $n_d \times m_2$ matrix. A general setup of the observed values can be presented as follows:

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_{n_c} \\ y_{n_c+1} \\ \vdots \\ y_{n_c+n_d} \end{bmatrix}, \qquad \mathbf{X}_1 = \begin{bmatrix} \mathbf{X}_1^c \\ \\ \mathbf{X}_1^d \\ \\ \end{bmatrix} = \begin{bmatrix} x_{11}^1 & \cdots & x_{1m_1}^1 \\ \vdots & \ddots & \vdots \\ x_{n_c1}^1 & \cdots & x_{n_cm_1}^1 \\ x_{(n_c+1)1}^1 & \cdots & x_{(n_c+1)m_1}^1 \\ \vdots & \ddots & \vdots \\ x_{(n_c+n_d)1}^1 & \cdots & x_{(n_c+n_d)m_1}^1 \end{bmatrix},$$

$$\mathbf{X}_2 = \begin{bmatrix} \mathbf{X}_2^c \\ \\ \mathbf{X}_2^d \\ \\ \end{bmatrix} = \begin{bmatrix} x_{11}^2 & \cdots & x_{1m_2}^2 \\ \vdots & \ddots & \vdots \\ x_{n_c1}^2 & \cdots & x_{n_cm_2}^2 \\ x_{(n_c+1)1}^2 & \cdots & x_{(n_c+1)m_2}^2 \\ \vdots & \ddots & \vdots \\ x_{(n_c+n_d)1}^2 & \cdots & x_{(n_c+n_d)m_2}^2 \end{bmatrix}.$$

In our package we propose 10 methods for testing interaction at the gene level. These 10

methods are all based on three main parameters: `Y`, a numeric or factor vector with exactly two distinct values, `G1` and `G2`, two '`SnpMatrix`' objects as proposed by the R **Bioconductor** package **snpStats**. '`SnpMatrix`' objects have been chosen because of their capacity to handle large datasets. Furthermore, '`SnpMatrix`' objects benefit from numerous methods provided by the **snpStats** package, allowing flexible data manipulation and efficient computation of summary statistics.

In the remainder of this section, we provide details on the methods implemented in our package and highlight the specific parameters used for each method. Our implementation is illustrated by the dataset `gene.pair` provided with the **GeneGeneInteR** package. The dataset `gene.pair` is a case-control dataset containing the genotypes of 8 SNPs within *GC* gene (object `G1` of class '`SnpMatrix`') and 4 SNPs within *PADI2* gene (object `G2` of class '`SnpMatrix`'). The dataset `gene.pair` further includes the disease status in a factor variable `Y`. In more details, `gene.pair` contains 247 individuals affected by rheumatoid arthritis (RA) (`Y = 1`) and 206 individuals not affected by RA (`Y = 0`).

The contents of the dataset `gene.pair` are summarized by the following command lines:

```
R> library("GeneGeneInteR")
R> data("gene.pair", package = "GeneGeneInteR")
R> summary(gene.pair)
```

```
     Length   Class      Mode
Y    453      factor     numeric
G1   3624     SnpMatrix  raw
G2   1812     SnpMatrix  raw
```

The next code line displays the header of the status coding variable `Y`:

```
R> head(gene.pair$Y)
```

```
[1] HealthControl HealthControl HealthControl HealthControl HealthControl
Levels: HealthControl RheumatoidArthritis
```

The two following code lines provide the main characteristics of the two '`SnpMatrix`' objects, `G1` and `G2`:

```
R> gene.pair$G1
```

```
A SnpMatrix with  453 rows and  8 columns
Row names:  Id1 ... Id453
Col names:  rs1491710 ... rs2298849
```

```
R> gene.pair$G2
```

```
A SnpMatrix with  453 rows and  4 columns
Row names:  Id1 ... Id453
Col names:  rs2057094 ... rs1005753
```

3.1. Principal component analysis (PCA)

In the PCA-based method, a likelihood ratio test is performed to compare the model $\mathcal{M}_{Inter}$ to the model $\mathcal{M}_{No}$, where $\mathcal{M}_{Inter}$ refers to the logistic model including interaction effects while $\mathcal{M}_{No}$ does not consider interaction terms. Formally, $\mathcal{M}_{Inter}$ is defined in Equation 1:

$$\text{logit}\left(\mathbb{P}\left[Y = 1|PC^1_{X_1}\dots PC^{n_1}_{X_1}, PC^1_{X_2}\dots PC^{n_2}_{X_2}\right]\right) = \beta_0 + \sum_{i=1}^{n_1} PC^i_{X_1} + \sum_{j=1}^{n_2} PC^j_{X_2} + \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} PC^i_{X_1} PC^j_{X_2} \quad (1)$$

and $\mathcal{M}_{No}$ in Equation 2:

$$\text{logit}\left(\mathbb{P}\left[Y = 1|PC^1_{X_1}\dots PC^{n_1}_{X_1}, PC^1_{X_2}\dots PC^{n_2}_{X_2}\right]\right) = \beta_0 + \sum_{i=1}^{n_1} PC^i_{X_1} + \sum_{j=1}^{n_2} PC^j_{X_2}. \quad (2)$$

In models $\mathcal{M}_{Inter}$ and $\mathcal{M}_{No}$, $PC^i_{X_1}$ and $PC^j_{X_2}$ are the ith principal component of $\mathbf{X}_1$ and the jth principal component of $\mathbf{X}_2$. The number of principal components, n_1 and n_2, kept in the interaction test is determined by the percentage of inertia retrieved by the PCA. Such a percentage is defined by the user and corresponds to the `threshold` parameter.

In our package, two distinct principal component decompositions are provided by the functions `PCA.test` via the argument `method`. With `method = "Std"`, the dataset is standardized using variables' standard deviation while with `method = "GenFreq"`, the dataset is standardized using standard deviation under Hardy-Weinberg equilibrium, as proposed in the **snpStats** package.

When the percentage of inertia asked by the user is high, the number of PCs can be important and fitting logistic models $\mathcal{M}_{Inter}$ and $\mathcal{M}_{No}$ is likely to fail. In that case, the number of PCs retained in model (2) is reduced until convergence of the `glm` function for fitting models $\mathcal{M}_{Inter}$ and $\mathcal{M}_{No}$.

The following code line illustrates the PCA-based method when the dataset is standardized using standard deviation under Hardy-Weinberg equilibrium:

```
R> PCA.test(Y = gene.pair$Y, G1 = gene.pair$G1, G2 = gene.pair$G2,
+    threshold = 0.7, method = "GenFreq")

     Gene-based interaction based on Principal Component Analysis - GenFreq

data:  gene.pair$Y and  (gene.pair$G1 , gene.pair$G2)
Deviance = 8.2157, df = 6.0, threshold = 0.7, p-value = 0.2227
alternative hypothesis: true deviance is greater than 0
sample estimates:
Deviance without interaction    Deviance with interaction
                    615.2977                     607.0821
```

In the next code line, the dataset is standardized using variables' standard deviation:

```
R> PCA.test(Y = gene.pair$Y, G1 = gene.pair$G1, G2 = gene.pair$G2,
+    threshold = 0.7, method = "Std)
```

```
     Gene-based interaction based on Principal Component Analysis - Std

data:  gene.pair$Y and  (gene.pair$G1 , gene.pair$G2)
Deviance = 8.5074, df = 6.0, threshold = 0.7, p-value = 0.2032
alternative hypothesis: true deviance is greater than 0
sample estimates:
Deviance without interaction    Deviance with interaction
                    615.0911                     606.5837
```

3.2. Canonical correlation analysis (CCA)

The CCA test is based on a Wald-type statistic defined as in the following Equation 3 (see Peng *et al.* 2010 for details):

$$U_{CCA} = \frac{z_d - z_c}{\sqrt{\mathbb{V}(z_d) + \mathbb{V}(z_c)}}, \tag{3}$$

where $z_d = \frac{1}{2}(\log(1+r_d) - \log(1-r_d))$ and $z_c = \frac{1}{2}(\log(1+r_c) - \log(1-r_c))$ with r_d the maximum canonical correlation coefficient between $\mathbf{X}_1^d$ and $\mathbf{X}_2^d$ and r_c the maximum canonical correlation coefficient between $\mathbf{X}_1^c$ and $\mathbf{X}_2^c$ computed for controls ($Y = 0$). As suggested by Peng *et al.* (2010), the variances $\mathbb{V}(z_d)$ and $\mathbb{V}(z_c)$ are determined by applying a bootstrapping method. The number of bootstrap samples used to estimate $\mathbb{V}(z_d)$ and $\mathbb{V}(z_c)$ is determined by the `n.boot` argument (with default `n.boot = 500`). p value is then obtained by noting that under the null hypothesis $U_{CCA} \sim \mathcal{N}(0,1)$.

The CCA based gene-gene interaction test is implemented in the `CCA.test` function and mainly depends on the `cancor` function from the **stats** package (R Core Team 2020).

The following code lines illustrate the use of the `CCA.test` function.

```
R> set.seed(1234)
R> CCA.test(Y = gene.pair$Y, G1 = gene.pair$G1, G2 = gene.pair$G2,
+    n.boot = 500)
```

```
     Gene-based interaction based on Canonical Correspondence Analysis

data:  gene.pair$Y and  (gene.pair$G1 , gene.pair$G2)
CCU = 0.60441, n.boot = 500, p-value = 0.5456
alternative hypothesis: true CCU is not equal to 0
sample estimates:
       z0        z1
0.2940799 0.2414700
```

3.3. Kernel canonical correlation analysis (KCCA)

The KCCA based test provides a generalization of the CCA test to detect non-linear co-association between $\mathbf{X}_1$ and $\mathbf{X}_2$ (Yuan *et al.* 2012; Larson and Schaid 2013) and is based on

the following Wald-type statistic:

$$U_{KCCA} = \frac{kz_d - kz_c}{\sqrt{\mathbb{V}(kz_d) + \mathbb{V}(kz_c)}}, \tag{4}$$

where $kz_d = \frac{1}{2}(\log(1 + kr_d) - \log(1 - kr_d))$ and $kz_c = \frac{1}{2}(\log(1 + kr_c) - \log(1 - kr_c))$ with kr_d the maximum kernel canonical correlation coefficient between $\mathbf{X}_1^d$ and $\mathbf{X}_2^d$ and kr_c the maximum kernel canonical correlation coefficient between $\mathbf{X}_1^c$ and $\mathbf{X}_2^c$.

Similar to the CCA test, $\mathbb{V}(kz_d)$ and $\mathbb{V}(kz_c)$ are estimated using bootstrap techniques (Yuan *et al.* 2012; Larson and Schaid 2013) and the p value is obtained using the standard Gaussian distribution of U_{KCCA} under the null hypothesis. Since the performance of kernel methods strongly relates to the choice of kernel functions, the default is the radial basis kernel function (RBF) owing to its flexibility in parameter specification. However, other kernel functions, such as linear, polynomial or spline kernels, can be used. Thus, in addition to the three arguments `Y`, `G1` and `G2`, our implementation of the KCCA test proposes two optional arguments: `n.boot` that determines the number of bootstrap samples and `kernel` that provides the kernel function to be used. This `kernel` parameter is a character string matching one of the kernel name provided by the **kernlab** package (Karatzoglou, Smola, Hornik, and Zeileis 2004) such as `"rbfdot"`, `"polydot"`, `"tanhdot"`, `"vanilladot"`, `"laplacedot"`, `"besseldot"`, `"anovadot"`, `"splinedot"`. The arguments, `sigma`, `degree`, `scale`, `offset` and `order`, can also be passed to the `kcca.test` function in order to parameterize the kernel used in the analysis.

The KCCA based gene-gene interaction test is implemented in the `KCCA.test` function and mainly depends on the `kcca` function from the **kernlab** package.

The next few lines give two examples of the use of the `KCCA.test` function. First, testing is performed by `KCCA.test` with a RBF kernel ($\sigma = 0.05$) and significance is tested according to 500 bootstrap replicates.

```
R> set.seed(1234)
R> KCCA.test(Y = gene.pair$Y, G1 = gene.pair$G1, G2 = gene.pair$G2,
+    kernel = "rbfdot", sigma = 0.05, n.boot = 500)

     Gene-based interaction based on Kernel Canonical Correspondence
     Analysis

data:  gene.pair$Y and  (gene.pair$G1 , gene.pair$G2)
KCCU = 0.0078595, n.boot = 500, p-value = 0.9937
alternative hypothesis: true KCCU is not equal to 0
sample estimates:
       z0        z1
-3.717346 -3.759154
```

In a second example, `KCCA.test` is used with a Bessel kernel (with hyperparameters `sigma` = 0.05, `degree` = 1 and `order` = 1) and significance is tested according to the default value of 100 bootstrap replicates.

```
R> set.seed(1234)
R> KCCA.test(Y = gene.pair$Y, G1 = gene.pair$G1, G2 = gene.pair$G2,
+    kernel = "besseldot", sigma = 0.05, degree = 1, order = 1, n.boot = 100)
```

```
     Gene-based interaction based on Kernel Canonical Correspondence
     Analysis

data:  gene.pair$Y and  (gene.pair$G1 , gene.pair$G2)
KCCU = 0.015163, n.boot = 100, p-value = 0.9879
alternative hypothesis: true KCCU is not equal to 0
sample estimates:
       z0        z1
-4.161048 -4.251702
```

3.4. Partial least square path modeling (PLSPM)

The PLSPM testing has been introduced by Zhang *et al.* (2013) and is based on the Wald-like statistic defined in Equation 5:

$$U_{PLSPM} = \frac{\beta_d - \beta_c}{\sqrt{\mathbb{V}(\beta_d - \beta_c)}} \tag{5}$$

where β_d (respectively, β_c) is the path coefficient between $\mathbf{X}_1^d$ and $\mathbf{X}_2^d$ (respectively, $\mathbf{X}_1^c$ and $\mathbf{X}_2^c$). As quoted by Zhang *et al.* (2013), the distribution of U_{PLSPM} is unknown and significance can be tested using permutations.

The PLSPM based gene-gene interaction test is implemented in the `PLSPM.test` function and mainly depends on the `plspm` function from the **plspm** package (Sanchez, Trinchera, and Russolillo 2017). Besides the three mandatory arguments `Y`, `G1` and `G2`, the number of permutations can be set by the `n.perm` argument (default is `n.perm = 500`).

The following code lines display an example of the use of the `PLSPM.test` function.

```
R> set.seed(1234)
R> PLSPM.test(Y = gene.pair$Y, G1 = gene.pair$G1, G2 = gene.pair$G2,
+    n.perm = 1000)

     Gene-based interaction based on Partial Least Squares Path Modeling

data:  gene.pair$Y and  (gene.pair$G1 , gene.pair$G2)
U = 4.0938, n.perm = 1000, p-value = 0.163
alternative hypothesis: true U is not equal to 0
sample estimates:
     beta0      beta1
-0.2269269  0.1474433
```

3.5. Composite linkage disequilibrium (CLD)

The CLD method, proposed in Rajapakse *et al.* (2012) is based on the normalized quadratic distance (NQD) and is defined in Equation 6:

$$\delta^2 = \text{tr.}\Big((\tilde{D} - \tilde{C})W^{-1}(\tilde{D} - \tilde{C})W^{-1}\Big), \tag{6}$$

where $\tilde{D}$, $\tilde{C}$ and W are three $(m_1 + m_2) \times (m_1 + m_2)$ matrices of the covariance between the whole set of SNPs that combines SNPs from both genes. More precisely, $\tilde{D}$ and $\tilde{C}$ are defined as follows in Equation 7:

$$\tilde{D} = \begin{bmatrix} W_{11} & D_{12} \\ D_{21} & W_{22} \end{bmatrix}, \qquad \tilde{C} = \begin{bmatrix} W_{11} & C_{12} \\ C_{21} & W_{22} \end{bmatrix}, \tag{7}$$

where W_{11} (respectively, W_{22}) is the pooled estimate of the covariance matrix for $\mathbf{X}_1$ (respectively, $\mathbf{X}_2$, $D_{12}(= D_{21}^\top)$ and $C_{12}(= C_{21}^\top)$ are the sample covariance matrix between the two genes estimated from $\left(\mathbf{X}_1^d, \mathbf{X}_2^d\right)$ and $(\mathbf{X}_1^c, \mathbf{X}_2^c)$ respectively. In more details, the sample covariance matrices in cases, denoted by D, and in controls, denoted by C, can be partitioned in 4 blocks as follows in Equation 8:

$$D = \mathrm{Cov}\left(\mathbf{X}_1^d, \mathbf{X}_2^d\right) = \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix}, \qquad C = \mathrm{Cov}\left(\mathbf{X}_1^c, \mathbf{X}_2^c\right) = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}. \tag{8}$$

The pooled estimate of the covariance matrix, W, can thus be obtained by Equation 9:

$$W = \frac{n_c C + n_d D}{n_c + n_d} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}. \tag{9}$$

Since the distribution of δ^2 is not known under the null hypothesis, significance testing is performed using permutation tests, as proposed by Rajapakse *et al.* (2012). Such a test has been implemented in our package in the `CLD.test` function where the number of permutations is determined by the argument `n.perm`.

In the next few lines, function `CLD.test` is illustrated with the `gene.pair` example.

```
R> set.seed(1234)
R> CLD.test(Y = gene.pair$Y, G1 = gene.pair$G1, G2 = gene.pair$G2,
+    n.perm = 2000)

     Gene-based interaction based on Composite Linkage Disequilibrium

data:  gene.pair$Y and  (gene.pair$G1 , gene.pair$G2)
CLD = 0.49257, n.perm = 2000, p-value = 0.889
alternative hypothesis: true CLD is not equal to 0
sample estimates:
      CLD
0.4925654
```

3.6. Gene-based information gain method (GBIGM)

Introduced by Li *et al.* (2015), the GBIGM method is based on the information gain rate $\Delta R_{1,2}$. $\Delta R_{1,2}$ is defined as in Equation 10:

$$\Delta R_{1,2} = \frac{\min(H_1, H_2) - H_{1,2}}{\min(H_1, H_2)}, \tag{10}$$

where H_1, H_2, $H_{1,2}$ are the conditional entropies (given $\mathbf{Y}$) of $\mathbf{X}_1$, $\mathbf{X}_2$ and the pooled SNP set $(\mathbf{X}_1, \mathbf{X}_2)$ respectively. Assuming that $H(\cdot)$ is the classical entropy function, we have:

$$H_1 = H(\mathbf{Y}, \mathbf{X}_1) - H(\mathbf{X}_1) \quad (11)$$
$$H_2 = H(\mathbf{Y}, \mathbf{X}_2) - H(\mathbf{X}_2) \quad (12)$$
$$H_{1,2} = H(\mathbf{Y}, \mathbf{X}_1, \mathbf{X}_2) - H(\mathbf{X}_1, \mathbf{X}_2) \quad (13)$$

Since the distribution of $\Delta R_{1,2}$ is unknown, the significance testing is performed by permutations as suggested by Li *et al.* (2015). The GBIGM method has been implemented in the `GBIGM.test` function and the number of permutations is defined by the argument `n.perm`.

The following code lines give an example of the `GBIGM.test` function.

```
R> set.seed(1234)
R> GBIGM.test(Y = gene.pair$Y, G1 = gene.pair$G1, G2 = gene.pair$G2,
+    n.perm = 2000)

     Gene-based interaction based on Gene-based Information Gain Method

data:  gene.pair$Y and  (gene.pair$G1 , gene.pair$G2)
DeltaR1,2 = 0.46441, n.perm = 2000, p-value = 0.449
alternative hypothesis: two.sided
sample estimates:
DeltaR1,2
0.4644093
```

3.7. From SNPxSNP interaction to gene-gene interaction testing

This section provides details of the four statistical methods that propose a gene-based test from SNP-based tests (Emily 2016). Rather than considering multiple SNPs in both genes as part of a joint model, these methods aim at aggregating p values obtained at the SNP level into a single p value at a gene level.

Interaction testing at the SNP level

Let consider a pair of SNPs, $(X_{1,j}, X_{2,k})$ where $X_{1,j}$ is the jth SNP of gene X_1 and $X_{2,k}$ the kth SNP of gene X_2 ($1 \leq j \leq m_1$ and $1 \leq k \leq m_2$). To test for interaction at the SNP level, we used the following Wald statistic defined in Equation 14:

$$W_{jk} = \frac{\widehat{\beta_3^{j,k}}}{\sigma\left(\widehat{\beta_3^{j,k}}\right)}, \quad (14)$$

where $\widehat{\beta_3^{j,k}}$ is an estimate of the interaction coefficient $\beta_3^{j,k}$ of the logistic model defined in Equation 15:

$$\log\left(\frac{\mathbb{P}[Y=1|X_{1,j}=x_1, X_{2,k}=x_2]}{1-\mathbb{P}[Y=1|X_{1,j}=x_1, X_{2,k}=x_2]}\right) = \beta_0^{j,k} + \beta_1^{j,k}x_1 + \beta_2^{j,k}x_2 + \beta_3^{j,k}x_1x_2. \quad (15)$$

$\widehat{\beta_3^{j,k}}$ is obtained by maximizing the likelihood function on the observed data $\mathbf{Y}$, $\mathbf{X}_1$ and $\mathbf{X}_2$, while $\sigma\left(\widehat{\beta_3^{j,k}}\right)$ is calculated by inverting the Hessian of the likelihood. Since the solution of the maximization of the likelihood function does not have a closed form, we compute W_{jk} according to the iteratively reweighted least squares algorithm proposed in the `glm` function of the **stats** package.

To combine the statistics W_{jk} into a single test, four methods have been proposed that all account for covariance matrix

$$\Sigma = \left[\sigma_{(j,k),(j',k')}\right]_{\substack{j=1\ldots m_1;k=1\ldots m_2\\ j'=1\ldots m_1;k'=1\ldots m_2}},$$

a $(m_1 \times m_2) \times (m_1 \times m_2)$ symmetric matrix where $\sigma_{(j,k),(j',k')} = \text{Cov}(W_{jk}, W_{j',k'})$ (Ma, Clark, and Keinan 2013). As proposed by Emily (2016), the covariance between W_{jk} and $W_{j',k'}$ is estimated by: $\widehat{\sigma_{(j,k),(j',k')}} = r_{j,j'} r_{k,k'}$ where $r_{j,j'} = \frac{p_{jj'} - p_j p_{j'}}{\sqrt{p_j(1-p_j)p_{j'}(1-p_{j'})}}$ is the widely used correlation measure between SNP j and SNP j', given that p_j and $p_{j'}$ are the respective allelic frequencies and $p_{jj'}$ is the joint allelic frequency (Hill and Robertson 1968).

minP

The minP test is based on the minimum p value that is often used to combine p values of association (Conneely and Boehnke 2007). Let $W_{\max} = \max\left(|W_{11}|, \ldots, |W_{m_1,m_2}|\right)$ be the maximum of the absolute observed statistics. The minP is then defined by:

$$\text{minP} = 1 - \mathbb{P}\Big[\max(|Z_1|, |Z_2|, \ldots, |Z_{m_1 m_2}|) < W_{\max}\Big], \tag{16}$$

where $\mathbb{Z} = (Z_1, Z_2, \ldots, Z_{m_1 m_2})$ is a random vector that follows a multivariate normal distribution $\mathbb{Z} \sim \mathcal{N}(\mathbf{0}, \Sigma)$.

The computation of Equation 16 requires the calculation of the probability distribution of a multivariate normal random variable. For that purpose, we used the `pmvnorm` function from the R package **mvtnorm** (Genz and Bretz 2009).

To illustrate the `minP` function, the following code lines test the interaction between the two sets of SNPs in the `gene.pair` example.

```
R> set.seed(1234)
R> minP.test(Y = gene.pair$Y, G1 = gene.pair$G1, G2 = gene.pair$G2)

     Gene-based interaction based on minP method

data:  gene.pair$Y and  (gene.pair$G1 , gene.pair$G2)
Wmax = 0.0099241, p-value = 0.1796
alternative hypothesis: true Wmax is greater than 0
sample estimates:
       Wmax
0.009924148
```

The function `pmvnorm` is applicable to dimensions up to 1000 (Genz and Bretz 2009). Thus, if we consider a first gene having m_1 SNPs and a second gene with m_2 SNPs, the minP test can

be performed only if $m_1 \times m_2 \leq 1000$. Furthermore, it has been shown that the computation of the minP test lacks accuracy when the number of tests is larger than 900 (Conneely and Boehnke 2007). To overcome such a limitation, we propose a two-steps approach to perform the minP procedure. In a first step, each gene (or set of SNPs) is divided into subsets of SNPs so that all pairs of subsets do not contain more than 900 SNP pairs. Formally, genes X_1 and X_2 are split in respectively k_1 and k_2 subsets as follows:

$$X_1 = [X_1^1, X_1^2, \ldots, X_1^{k_1}], \qquad X_2 = [X_2^1, X_2^2, \ldots, X_2^{k_2}].$$

The number of subsets per gene (namely k_1 and k_2) are chosen to ensure that $\forall i \in \{1, \ldots, k_1\} : |X_1^i| \leq 30$ and $\forall j \in \{1, \ldots, k_2\} : |X_2^j| \leq 30$, where $|X_1^i|$ and $|X_2^j|$ are the number of SNPs in the subset i from gene X_1 and in the subset j from gene X_2. By doing so, we guarantee that:

$$\forall i \in \{1, \ldots, k_1\}; j \in \{1, \ldots, k_2\} : |X_1^i| \times |X_2^j| \leq 900.$$

To keep the 1-dimensional structure of the data, each gene is split according to a constraint hierarchical clustering based on the pairwise LD between SNPs. To perform such a clustering, we implemented in our **GeneGeneInteR** package, the `chclust` function.

In a second step, the `minP` procedure is applied to each pair of subsets to obtain a vector of $k_1 \times k_2$ p values: $[p_{11}, \ldots, p_{ij}, \ldots, p_{k_1,k_2}]$, where p_{ij} is the p value given by the minP test based on subsets X_1^i and X_2^j. To correct for multiple testing, a Benjamini-Hochberg correction is applied to the vector $[p_{11}, \ldots, p_{ij}, \ldots, p_{k_1,k_2}]$ and the minimum of the corrected p values is returned by the `minP.test` function.

To illustrate such a strategy, consider the interaction between two genes *PCSK6* and *TXNDC5*, where *PCSK6* (respectively, *TXNDC5*) contains 54 (respectively, 36) SNPs. The number of SNPxSNP interaction tests is therefore equal to $54 \times 36 = 1944 > 1000$, thus preventing the direct use of the function `pmvnorm`. Therefore, in a first step, constraint hierarchical trees are estimated for each gene and then used to split each gene into SNP subsets. As shown in Figure 2, gene *PCSK6* is split into $k_1 = 4$ subsets and gene *TXNDC5* is divided into $k_2 = 2$ subsets. It can be remarked that the sizes of the subsets for gene *PCSK6* are 19, 2, 12 and 21 while equal to 10 and 26 for gene *TXNDC5*. Therefore, the maximum number of SNPxSNP interaction tests between the two subsets is $33 \times 26 = 858$ which is lower than 900. In a second step, each pair of subsets is tested for interaction leading to a vector of six p values $[p_{11} = 0.11, p_{21} = 0.87, p_{31} = 0.21, p_{41} = 0.05, p_{12} = 0.30, p_{22} = 0.10, p_{32} = 0.78, p_{42} = 0.30]$. These six p values are combined into a single p value by taking the minimum of Benjamini-Hochberg corrected p values. The overall p value is therefore given by 0.2925908.

These two steps are implemented in the `minP.test` function and are invisible to the user. The following code lines detail the output of the `minP.test` function when testing for interaction between the two genes *PCSK6* and *TXNDC5*. We start by importing the disease status `Y`:

```
R> resp <- system.file(file.path("extdata", "response.txt"),
+    package = "GeneGeneInteR")
R> Y <- read.csv(resp, header = FALSE, stringsAsFactors = TRUE)
```

With the three following lines, the two sets of SNPs (i.e., the two genes *PCSK6* and *TXNDC5*) are imported:

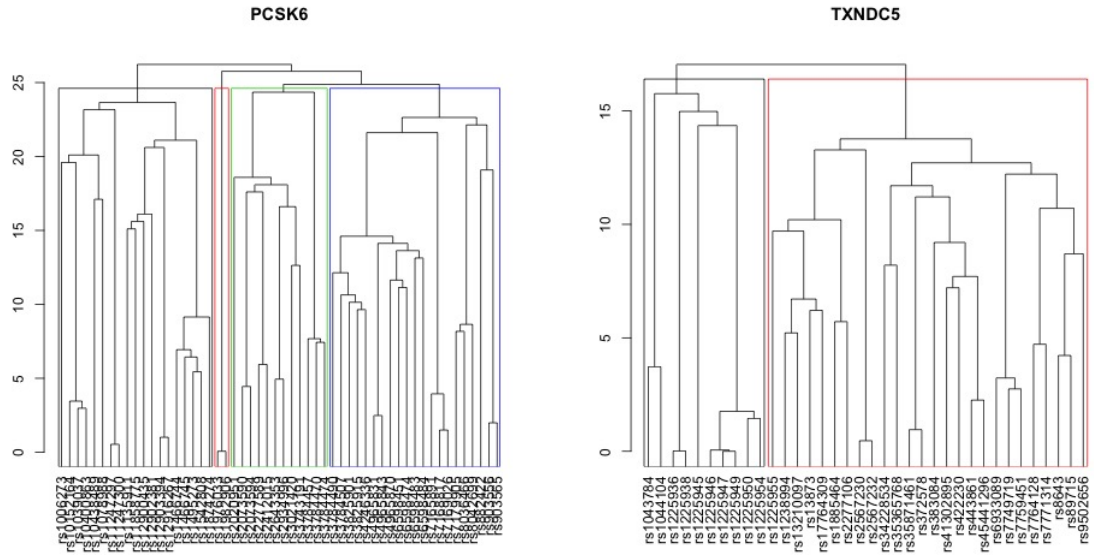

Figure 2: Constraint hierarchical trees obtained for genes *PCSK6* and *TXNDC5*. The rectangles correspond to the subsets of SNPs estimated by our proposed strategy.

```
R> load(system.file(file.path("extdata", "dataImputed.Rdata"),
+    package = "GeneGeneInteR"))
R> PCSK6 <- selectSnps(data.imputed$snpX, data.imputed$genes.info,
+    "PCSK6")$snpX
R> TXNDC5 <- selectSnps(data.imputed$snpX, data.imputed$genes.info,
+    "TXNDC5")$snpX
```

We can then check the number of SNPs that belong to each gene:

```
R> ncol(PCSK6)

[1] 54

R> ncol(TXNDC5)

[1] 36
```

Finally, the testing of the interaction between these two genes can be performed directly with the `minP.test` function as follows:

```
R> set.seed(1234)
R> minP.test(Y = Y, G1 = PCSK6, G2 = TXNDC5)

     Gene-based interaction based on minP method

data:  Y and  (PCSK6 , TXNDC5)
```

```
Wmax = 0.00093876, p-value = 0.2926
alternative hypothesis: greater
sample estimates:
        Wmax
0.0009387614
```

GATES

The GATES procedure, proposed by Li *et al.* (2011), is an extension of the Simes procedure used to assess the gene level association significance. Let $p_{(1)}, \ldots, p_{(m_1 m_2)}$ be the ascending SNPxSNP interaction $m_1 \times m_2$ p values, GATES p value is then defined in Equation 17:

$$\mathrm{p}_{GATES} = \min\left(p_{(1)}\frac{me}{me_{(1)}}, p_{(2)}\frac{me}{me_{(2)}}, \ldots, p_{(m_1 m_2)}\frac{me}{me_{(m_1 m_2)}}\right), \tag{17}$$

where me is the number of effective tests among the $m_1 \times m_2$ tests and $me_{(i)}$ the number of effective tests among the i most significant tests associated with the lowest order p values $p_{(1)}, \ldots, p_{(i)}$. The number of effective tests ought to characterize the number of independent tests equivalent to the correlated tests that are really performed and is often used to account for dependence in a multiple testing correction.

Although no formal definition of the number of effective tests has been formulated in the literature, several procedures have been proposed to estimate such number. All methods are based on a transformation of the set of eigenvalues of the SNP covariance matrix assuming that (1) if the SNPs are independent, the number of effective tests is the number of performed tests, (2) if the absolute value of the correlation between any pair of SNPs is equal to 1, the number of effective tests is 1. In the **GeneGeneInteR** package, four main methods have been implemented within the `gates.test` function and can be chosen by the user with the argument `me.est`: Cheverud-Nyholt method – `me.est = "ChevNy"` (Cheverud 2001; Nyholt 2004), Keff method – `me.est = "Keff"` (Moskvina and Schmidt 2008), Li and Ji method – `me.est = "LiJi"` (Li and Ji 2005) and Galwey – `me.est = "Galwey"` (Galwey 2009).

The following code lines give examples of the use of the `gates.test` function with all estimated methods for the number of effective tests:

```
R> set.seed(1234)
R> gates.test(Y = gene.pair$Y, G1 = gene.pair$G1, G2 = gene.pair$G2,
+    me.est = "ChevNy")

     Gene-based interaction based on GATES method

data:  gene.pair$Y and  (gene.pair$G1 , gene.pair$G2)
GATES = 0.0099241, p-value = 0.2939
alternative hypothesis: less
sample estimates:
      GATES
0.009924148

R> set.seed(1234)
R> gates.test(Y = gene.pair$Y, G1 = gene.pair$G1, G2 = gene.pair$G2,
+    alpha = 0.05, me.est = "Keff")
```

```
     Gene-based interaction based on GATES method

data:  gene.pair$Y and  (gene.pair$G1 , gene.pair$G2)
GATES = 0.013945, p-value = 0.1899
alternative hypothesis: less
sample estimates:
     GATES
0.01394543

R> set.seed(1234)
R> gates.test(Y = gene.pair$Y, G1 = gene.pair$G1, G2 = gene.pair$G2,
+    me.est = "LiJi")

     Gene-based interaction based on GATES method

data:  gene.pair$Y and  (gene.pair$G1 , gene.pair$G2)
GATES = 0.013945, p-value = 0.1255
alternative hypothesis: less
sample estimates:
     GATES
0.01394543

R> set.seed(1234)
R> gates.test(Y = gene.pair$Y, G1 = gene.pair$G1, G2 = gene.pair$G2,
+    me.est = "Galwey")

     Gene-based interaction based on GATES method

data:  gene.pair$Y and  (gene.pair$G1 , gene.pair$G2)
GATES = 0.013945, p-value = 0.1596
alternative hypothesis: less
sample estimates:
     GATES
0.01394543
```

tTS and tProd

The tTS and tProd procedures are two truncated tail strength methods that aim at combining signals from all single-SNP p values less than a predefined cutoff value (Jiang *et al.* 2011). Denoting by τ the cutoff value, the two truncated p values are defined as follows (Zaykin *et al.* 2002):

$$tTS = \frac{1}{m_1 m_2} \sum_{i=1}^{m_1 m_2} \mathbb{I}(p_{(i)} < \tau) \left(1 - p_{(i)} \frac{m_1 m_2 + 1}{i}\right), \tag{18}$$

$$tProd = \prod_{i=1}^{m_1 m_2} p_i^{\mathbb{I}(p_i < \tau)}, \tag{19}$$

where $\mathbb{I}$ is the indicator function. When p values are correlated, the null distributions of tTS and $tProd$ are unknown. Following the approach proposed by Zaykin *et al.* (2002), a p value is obtained in the **GeneGeneInteR** package by computing an empirical null distribution using Monte Carlo (MC) simulations. For each MC iteration, an empirical value for tTS (or $tProd$) is obtained by simulating a vector of W_{jk} with respect to a multivariate normal distribution with a vector of 0 means and $\widehat{\Sigma}$ as covariance matrix. The empirical p value is calculated as the proportion of simulated statistics larger than the observed statistic on the "true" set of W_{jk}.

The tTS and tProd methods have been implemented in the functions `tTS.test` and `tProd.test` of the **GeneGeneInteR** package. Additional to the mandatory `Y`, `G1` and `G2` arguments, these two functions have two optional arguments: `tau` and `n.sim` that control the cutoff value and the number of simulations used to estimate the empirical value respectively. The following code lines give an example of the use of the `tTS.test` and `tProd.test` functions:

```
R> set.seed(1234)
R> tTS.test(Y = gene.pair$Y, G1 = gene.pair$G1, G2 = gene.pair$G2,
+    tau = 0.5, n.sim = 10000)

     Gene-based interaction based on the Truncated Tail Strength method

data:  gene.pair$Y and  (gene.pair$G1 , gene.pair$G2)
tTS = -0.0099127, tau = 0.5, p-value = 0.5104
alternative hypothesis: less
sample estimates:
         tTS
-0.009912706

R> set.seed(1234)
R> tProd.test(Y = gene.pair$Y, G1 = gene.pair$G1, G2 = gene.pair$G2,
+    tau = 0.05, n.sim = 1000)

     Gene-based interaction based on the Truncated Product method

data:  gene.pair$Y and  (gene.pair$G1 , gene.pair$G2)
tProd = 0.0001384, tau = 0.05, p-value = 0.265
alternative hypothesis: less
sample estimates:
       tProd
0.0001383965
```

4. Analysis of a set of genes

In practice, it is very common to investigate the interaction between a collection of $k > 2$ genes. In this context, our package **GeneGeneInteR** proposes several methods to perform a complete pipeline of analysis, going from data importation to results visualization via data

manipulation and statistical analysis. In the remainder of this section, we illustrate our pipeline through the analysis of a case-control dataset publicly available in the NCBI repository GSE39428 series (Chang, Xu, Wang, Wang, Wang, and Yan 2013). The dataset contains the genotypes of 312 SNPs from 17 genes in a total of 429 patients (266 individuals affected by theumatoid arthritis and 163 healthy controls) and is attached to our package **GeneGeneInteR** as an external file in PED format supported by the **PLINK** software (Purcell *et al.* 2007).

In the following, the description of the complete statistical pipeline is decomposed into four main parts. In Section 4.1, we first describe the importation of the genotype and the phenotype data as proposed in our **GeneGeneInteR** package. Then, in Section 4.2, we focus on the various functions dedicated to the manipulation of the genotype data. In Section 4.3, we detail the statistical analysis performed on a set of genes. Finally, in Section 4.4, we introduce two main functionalities for visualizing the obtained results.

4.1. Genotype and phenotype importation

At first, the path for the files containing genotype data and information regarding the SNP set are loaded.

Genotype importation

```
R> ped <- system.file(file.path("extdata", "example.ped"),
+    package = "GeneGeneInteR")
R> info <- system.file(file.path("extdata", "example.info"),
+    package = "GeneGeneInteR")
R> posi <- system.file(file.path("extdata", "example.txt"),
+    package = "GeneGeneInteR")
```

The importation is performed with the `importFile` function:

```
R> data <- importFile(file = ped, snps = info, pos = posi, pos.sep = "\t")
```

The object `data` is a list of 3 elements: `status`, `snpX` (a '`SnpMatrix`' object) and `genes.info` (a '`data.frame`'). The `status` is available only if the imported format is PED.

```
R> summary(data)

           Length Class      Mode
status        429 factor     numeric
snpX       133848 SnpMatrix  raw
genes.info      4 data.frame list
```

We can check that the `snpX` object contains the genotype of 429 individuals for 312 SNPs.

```
R> data$snpX

A SnpMatrix with  429 rows and  312 columns
Row names:  H97 ... RA345
Col names:  rs1002788 ... rs9502656
```

The `genes.info` is a '`data.frame`' with exactly four columns that are named as follows: `Chromosome`, `Genenames`, `SNPnames` and `Position`.

```
R> summary(data$genes.info)
```

```
   Chromosome       Genenames        SNPnames      Position
 Min.   : 1.000   PCSK6  :74   rs1002788 :  1   Min.   :  7881078
 1st Qu.: 6.000   TXNDC5 :69   rs1005753 :  1   1st Qu.: 11712674
 Median : 8.000   DNAH9  :41   rs1006273 :  1   Median : 47863803
 Mean   : 9.962   CA1    :38   rs10152164:  1   Mean   : 52968484
 3rd Qu.:15.000   VDR    :19   rs10184179:  1   3rd Qu.: 97191582
 Max.   :17.000   Gc     :12   rs1032551 :  1   Max.   :123157722
                  (Other):59   (Other)   :306
```

Case-control status importation

Similar to functions introduced to analyze a single pair of genes (see Section 3), the case-control status is stored in a numeric or a factor vector with exactly two distinct values. If the phenotype is saved in a separate file in table form, it can thus be imported simply by using the `read.table` function such as:

```
R> Y <- read.table(system.file(file.path("extdata", "response.txt"),
+    package = "GeneGeneInteR"), sep = ";")
```

If the case-control status is provided in the PED file, it can be obtained as follows:

```
R> Y <- data$status
```

4.2. Genotype data manipulation

Once genotype data have been imported, SNPs can first be selected in a filtering step in order to improve the quality of the data or to focus on particular genomic regions. Furthermore, we developed functions to allow the imputation of missing genotypes. These two parts are detailed in the following paragraphs.

Data filtering

Before performing the statistical analysis, it is very common to remove some SNPs in order to improve the quality of the data. Such a cleaning step can be performed in our **GeneGeneInteR** package by using the function `snpMatrixScour`. `snpMatrixScour` aims at modifying a '`SnpMatrix`' object by removing SNPs that do not meet criteria regarding the minor allele frequency (MAF), deviation to the Hardy-Weinberg equilibrium (HWE) and the proportion of missing values. In the following example, SNPs with MAF lower than 0.05 or SNPs with p value for HWE lower than 0.001 or SNPs with a call rate lower than 0.9 are removed from the object `data`.

```
R> data <- snpMatrixScour(data$snpX,genes.info = data$genes.info,
+    min.maf = 0.05, min.eq = 1e-3, call.rate = 0.9)
```

```
A list object has been returned with elements: snpX & genes.info
```

The following code lines show that the dataset now contains only 209 SNPs, meaning that 103 SNPs have been filtered out.

```
R> data$snpX

A SnpMatrix with  429 rows and  209 columns
Row names:  H97 ... RA345
Col names:  rs10510123 ... rs4328262
```

Since the use of stringent filters could lead to the elimination of all SNPs within a gene, care has to be taken during the filtering step. However, in such a situation a gene without SNPs is removed from the dataset and a warning message is provided for the user.

In other situations, the user might be interested in performing the analysis on a predefined subset of SNPs. For that purpose, the `selectSnps` function provides three options to extract a collection of SNPs by specifying the argument `select` that should be one of the following:

- a numeric vector with only the column number in the '`snpMatrix`' (or row number for `genes.info`) of each selected SNP. The following code line allows the extraction of the 10 first SNPs:

  ```
  R> selec <- selectSnps(data$snpX, data$genes.info, select = 1:10)
  ```

- a character vector with the names of each selected SNP or each selected gene. The following example is used to extract genes *DNAH9* and *TXNDC5*:

  ```
  R> selec <- selectSnps(data$snpX, data$genes.info, c("DNAH9", "TXNDC5"))
  ```

- a character vector which elements are position bounds of genes. Each element of the vector is either of the form `"begin:end"`, or `"chr:begin:end"` if you have to specify the chromosome of the gene. The following code allows to select SNPs from position 101342000 to 101490000 on chromosome 15:

  ```
  R> selec <- selectSnps(data$snpX, data$genes.info,
  +    c("15:101342000:101490000"))
  ```

Genotype imputation

Since our pipeline of analysis does not handle missing values, SNPs filtering as well as SNPs selection can help removing missing data. As described in the previous section, this can be done easily by applying the `snpMatrixScour` function with argument `call.rate = 1`. However, in this case, SNPs with an acceptable call rate are also removed and the lost information is likely to be critical. Genotype imputation is then commonly performed to keep most of the informative SNPs in the dataset. Since our genotype data are stored into a '`SnpMatrix`' object, we implement the `imputeSnpMatrix` function that wraps functions `snp.imputation` and `impute.snps` from package **snpStats**. Our `imputeSnpMatrix` function mimics a leave-one-out process where missing SNPs are imputed for an individual based on a model trained on all other individuals.

In our example, the following code lines show that after the filtering step, 844 missing values still remain in the dataset.

```
R> sum(is.na(data$snpX))

[1] 844
```

To impute those missing values, we used our `imputeSnpMatrix` function as follows:

```
R> data <- imputeSnpMatrix(data$snpX, data$genes.info)

  |------------------------------------------------------------------| 100%
```

A simple check of the dataset shows that all missing values have been imputed:

```
R> sum(is.na(data$snpX))

[1] 0
```

When the amount of missing values is so large that `snp.imputation` is not able to find a rule of imputation, some missing values may remain. In this case, the user can specify the action to be done thanks to the `om.rem` argument:

- `om.rem = "none"`: leave the dataset as it is,
- `om.rem = "SNP"`: remove all SNPs with remaining missing values,
- `om.rem = "ind"`: remove all individuals with remaining missing values.

It is noteworthy that removing all SNPs is often more parsimonious than removing individuals and allows to get a dataset without any missing values with minimum information loss.

Although, function `snp.imputation` can calculate accurate rules for imputation, we encourage the user to first impute missing genotypes with an external software (such as **IMPUTE2**; Howie, Donnelly, and Marchini 2009) prior to the importation step.

4.3. Statistical analysis

The statistical analysis of a set of genes, as implemented in the `GGI` function, consists in performing all possible pairwise tests between two genes. Pairwise tests are conducted by using the `method` argument with one of the 10 methods detailed in Section 3. The `GGI` function takes two further mandatory arguments: `Y` the vector of case-control status and `snpX`, a '`SnpMatrix`' object that stores the genotypes for all SNPs. It is assumed that SNPs within the same gene are consecutive in the `snpX` argument. Furthermore, gene information, such as gene ordering and the number of SNPs within each gene, has to be provided either in the `genes.length` or in the `gene.info` argument.

The following code line allows the computation of all pairwise tests between the 17 genes of our example dataset with the PCA-based method.

```
R> GGI.res <- GGI(Y = Y, snpX = data$snpX, genes.info = data$genes.info,
+    method = "PCA")
```

The output of the GGI function is a an object of class 'GGInetwork'.

```
R> class(GGI.res)

[1] "GGInetwork"
```

The class 'GGInetwork' is an S3 class based on a list of 4 elements statistic, p.value, method and parameter. When method = "PCA", a fifth element, called df, is added to the 'GGInetwork'.

```
R> names(GGI.res)

[1] "statistic" "p.value"   "df"        "method"    "parameter"
```

Elements statistic, p.value and statistic, df are square matrices with M rows and M columns where M is the number of genes in the dataset. The elements of each matrix are the statistic, the p value and the degrees of freedom of the likelihood ratio test, respectively. The element method is the name of the method used to perform the pairwise interaction tests. Finally, the element parameter is a list of the parameters used to perform the pairwise interaction tests.

As example, the GGI.res object generated in the previous example is a list of 5 elements. Each cell of the output p.value matrix is the p value of the corresponding pairwise test. The pairwise p values obtained for the 4 first genes (*bub3*, *CA1*, *CDSN*, *DNAH9*) of our dataset can be observed as follows:

```
R> round(GGI.res$p.value[1:4, 1:4], digits = 4)

        bub3    CA1   CDSN  DNAH9
bub3  0.0000 0.1684 0.3179 0.1851
CA1   0.1684 0.0000 0.0697 0.0000
CDSN  0.3179 0.0697 0.0000 0.4539
DNAH9 0.1851 0.0000 0.4539 0.0000
```

Significant results can be summarized using the S3 method summary for class 'GGInetwork'. The method summary prints out the pairs of genes with an interaction p value lower than 0.05 after (1) no correction (2) a Bonferroni correction and (3) a Benjamini & Hochberg correction for multiple testing.

```
R> summary(GGI.res)

Gene-gene interaction network of  17  genes performed with:
          PCA

Significant interaction with no correction at the level of 0.05
```

```
-------
      Gene1    Gene2 Uncorrected p-value
1    TXNDC5      VDR             8.9e-10
2     DNAH9   TXNDC5             1.9e-09
[...]

Significant interaction with a bonferroni correction at the level of 0.05
-------
    Gene1    Gene2 bonferroni p-value
1  TXNDC5      VDR            1.2e-07
2   DNAH9   TXNDC5            2.6e-07
[...]

Significant interaction with a Benjamini & Hochberg correction at the
level of 0.05
-------
    Gene1    Gene2 BH p-value
1  TXNDC5      VDR    1.2e-07
2   DNAH9   TXNDC5    1.3e-07
[...]
```

4.4. Visualization

The results are visualized with the S3 method `plot` for class '`GGInetwork`'. Given an object of class '`GGInetwork`' obtained from the analysis of M genes with our `GGI` function, results can be visualized through two types of representation: an heatmap-like visualization with the `method = "heatmap"` argument and a network-like representation with the `method = "network"` argument.

Heatmap-like visualization

The `plot` method can be used with the '`GGInetwork`' object as the single input argument. Figures 3(a) and (b) show the graphical representation where all pairwise interactions are plotted (Figure 3(a) created with `plot(GGI.res)`) or only the interaction between the 3 genes *CA1*, *Gc* and *PADI1* with the argument `genes` (Figure 3 (b) created with `plot(GGI.res, genes = c("CA1", "Gc", "PADI1"))`).

When the number of genes is below 15, p values and names are drawn to make matrix reading easier (see Figure 3(b)). However, when the number of genes is larger than 15, p values are not drawn and if the number of genes is even larger than 25, none of the p values or the gene names are displayed (see Figure 3(a)). In this case, by setting the argument `interact = TRUE`, the user can start an interactive process that allows to click on a cell of interest to open a tooltip displaying which genes are involved in the selected interaction and the p value of the interaction test. Tooltips can be closed if the user clicks anywhere else than on a cell. This process stops when the user presses the escape button (or terminates the locator procedure in general) or when the user clicks on any place other than a cell when no tooltip window is open.

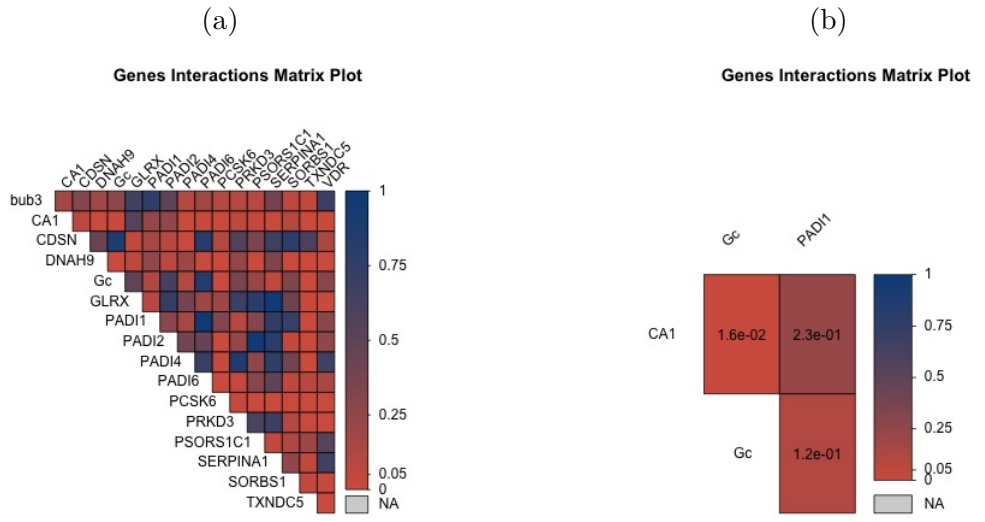

Figure 3: Default output of the `plot` method for '`GGInetwork`' objects considering (a) the whole set of genes or (b) a restricted set of 3 genes.

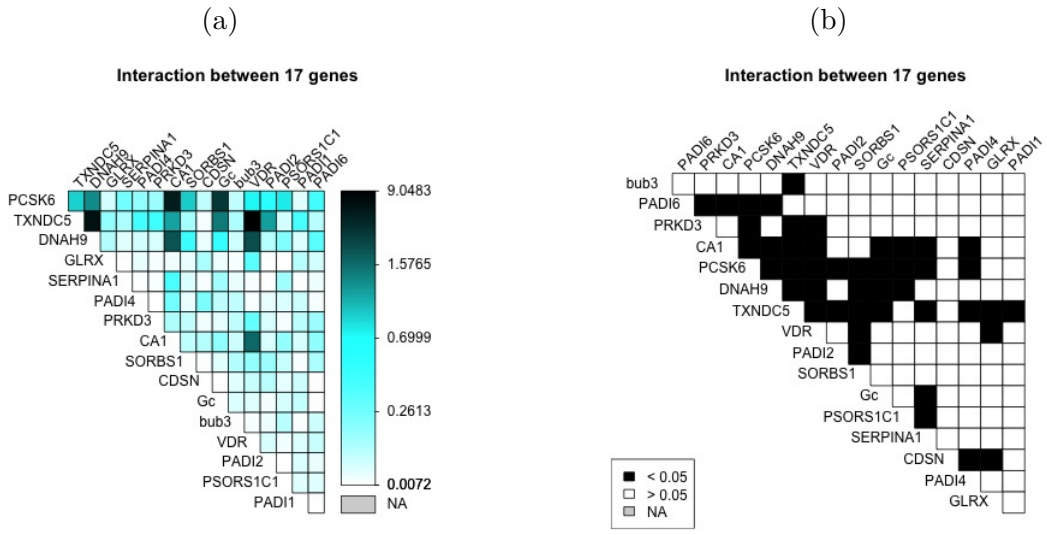

Figure 4: Example of the use of `plot` arguments (a) when no threshold is applied to the p values and (b) when a threshold of 0.05 is applied to the p values.

Several arguments can further be specified to customize the output graphics such as colors (arguments `col` and `NA.col`), width of the bar for colors (argument `colbar.width`), and titles (argument `title`). Users can also decide whether p values (argument `draw.pvals`) and gene names (argument `draw.names`) should be drawn and they may disable the interactivity of the plot (argument `interact`). To further improve plot clarity and hence allowing a better interpretation of the results, (1) genes can be ordered according to a hierarchical clustering (argument `hclust.order`), (2) p values can be reported in $-\log10$ scale (argument `use.log`) and (3) a threshold can be applied to the p values in order to distinguish between significant and non-significant interactions (argument `threshold`).

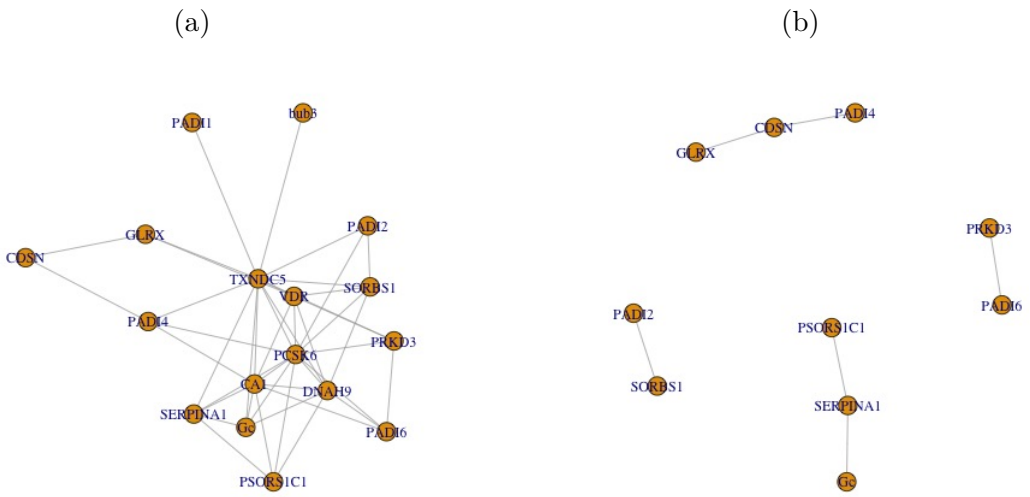

Figure 5: Output of the `plot` function used with (a) the argument `method = "network"` and (b) argument `method = "network"` and other customized arguments.

Figure 4 provides two plots resulting from different sets of arguments passed to the `plot` function:

```
R> plot(GGI.res, col = c("black", "cyan", "white"), colbar.width = 0.25,
+    title = "Interaction between 17 genes", hclust.order = TRUE,
+    use.log = TRUE, threshold = NULL, NA.col = "#D3D3D3",
+    draw.pvals = FALSE, draw.names = TRUE, interact = FALSE)
R> plot(GGI.res, col = c("black", "cyan", "white"), colbar.width = 0.05,
+    title = "Interaction between 17 genes", hclust.order = TRUE,
+    use.log = FALSE, threshold = 0.05, NA.col = "#D3D3D3",
+    draw.pvals = FALSE, draw.names = TRUE, interact = FALSE)
```

Network-like visualization

The `plot` function with `method = "network"` aims at drawing a graph between genes where two genes are adjacent if the p values between these two genes is below a given threshold (argument `threshold` with a default value equal to 0.05). The display of the network is performed by utilizing the `graph_from_data_frame` function from the **igraph** R package.

Two additional arguments can be used to customize the network. First, user can focus on a specific subset of genes with the argument `genes` and genes not linked to other genes can be removed from the graph with argument `plot.nointer`.

Figure 5(a) is created using

```
R> set.seed(1234)
R> plot(GGI.res, method = "network")
```

and displays the default network obtained with all genes. In Figure 5(b), a subset of only 12 genes have been selected to be the vertices of the graph using

```
R> set.seed(1234)
R> plot(GGI.res, method = "network", genes = c("bub3", "CDSN", "Gc", "GLRX",
+    "PADI1", "PADI2", "PADI4", "PADI6", "PRKD3", "PSORS1C1", "SERPINA1",
+    "SORBS1"), threshold = 0.05, plot.nointer = FALSE)
```

However, genes *bub3* and *PADI1* do not have a p value below the threshold of 0.05 with any of the other selected genes. Since the argument `plot.nointer` is set to `TRUE`, the two genes *bub3* and *PADI1* are not drawn in the resulting network.

5. Discussion

This article presents the R package **GeneGeneInteR** dedicated to the detection of an interaction between SNP sets in case-control genome-wide association studies. The package includes the `GGI` function performing all pairwise tests between two SNP sets and producing output of the class '`GGInetwork`'. The `GGI` function takes two main arguments: `method` and `snpX`, a '`SnpMatrix`' object. The `method` argument allows the user to specify the statistical procedure used for pairwise interaction testing as one of the ten methods implemented in **GeneGeneInteR**. Furthermore, several methods have been implemented in **GeneGeneInteR** to import, manipulate and impute missing values for the `snpX` objects. Methods `summary` and `plot`, associated with objects of the class '`GGInetwork`', are also proposed. The `plot` method allows the production of heatmap or network visualization of the statistical interaction between SNP sets. The R package **GeneGeneInteR** is available from **Bioconductor** (https://bioconductor.org/packages/GeneGeneInteR/) and from GitHub (https://github.com/MathieuEmily/GeneGeneInteR) .

Since our implementation relies on '`SnpMatrix`' objects, **GeneGeneInteR** could in principle handle large datasets. However, in practice, testing for pairwise interaction remains a combinatorial burden and highly computationally efficient methods are needed to perform large-scale analysis. In **GeneGeneInteR**, some methods are still computationally intensive (KCCA, PLSPM, GBIGM for example) so that their use is not recommended for large datasets. Possible extensions of **GeneGeneInteR** could consist in speeding up the execution time either by using parallel processing with the package **parallel** (R Core Team 2020) or by porting existing R code to C++ with **Rcpp** (Eddelbuettel and François 2011). Another limitation raised by the analysis of large scale datasets is the visualization of the results. However, we proposed several options to the user in order to focus on specific parts of a big network. Furthermore, the visualization of a big network provides information regarding a global pattern of relationships between SNP sets. Since the main purpose of the package **GeneGeneInteR** is to search for pairwise interactions, the detection of specific patterns in a network is beyond the scope of this work.

Finally, one of the main advantages of the **GeneGeneInteR** package lies in the use of a common pipeline of analysis for a collection of statistical procedures. Thus, **GeneGeneInteR** can easily be extended with the inclusion of novel statistical procedures to detect pairwise interaction between SNP sets. Further extensions could include the possibility to search for high-order of interactions (such as three-way interaction) between SNP sets.

References

Aulchenko YS, Ripke S, Isaacs A, van Duijn CM (2007). "**GenABEL**: An R Library for Genome-Wide Association Analysis." *Bioinformatics*, **23**(10), 1294–1296. doi:10.1093/bioinformatics/btm108.

Bhattacharjee S, Chatterjee N, Han S, Song M, Wheeler W (2019). ***CGEN**: An R Package for Analysis of Case-Control Studies in Genetic Epidemiology*. doi:10.18129/B9.bioc.CGEN. R package version 3.23.0.

Burkett K, Graham J, McNeney B (2006). "**hapassoc**: Software for Likelihood Inference of Trait Associations with SNP Haplotypes and Other Attributes." *Journal of Statistical Software*, **16**(2), 1–19. doi:10.18637/jss.v016.i02.

Chang X, Xu B, Wang L, Wang Y, Wang Y, Yan S (2013). "Investigating a Pathogenic Role for TXNDC5 in Tumors." *International Journal of Oncology*, **43**(6), 1871–1884. doi:10.3892/ijo.2013.2123.

Cheverud JM (2001). "A Simple Correction for Multiple Comparisons in Interval Mapping Genome Scans." *Heredity*, **87**(1), 52–58. doi:10.1046/j.1365-2540.2001.00901.x.

Clayton D (2020). ***snpStats**: 'SnpMatrix' and 'XSnpMatrix' Classes and Methods*. doi:10.18129/B9.bioc.snpStats. R package version 1.38.0.

Conneely KN, Boehnke M (2007). "So Many Correlated Tests, So Little Time! Rapid Adjustment of p Values for Multiple Correlated Tests." *The American Journal of Human Genetics*, **81**(6), 1158–1168. doi:10.1086/522036.

Csardi G, Nepusz T (2006). "The **igraph** Software Package for Complex Network Research." *InterJournal*, **Complex Systems**, 1695.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Emily M (2016). "AGGrEGATOr: A Gene-Based GEne-Gene interActTiOn Test for Case-Control Association Studies." *Statistical Applications in Genetics and Molecular Biology*, **15**(2), 151–171. doi:10.1515/sagmb-2015-0074.

Emily M, Sounac N, Kroell F, Houee-Bigot M (2020). ***GeneGeneInteR**: Tools for Testing Gene-Gene Interaction at the Gene Level*. doi:10.18129/B9.bioc.GeneGeneInteR. R package version 1.15.1.

Galwey NW (2009). "A New Measure of the Effective Number of Tests, a Practical Tool for Comparing Families of Non-Independent Significance Tests." *Genetic Epidemiology*, **33**(7), 559–568. doi:10.1002/gepi.20408.

Gentleman RC, Carey VJ, Bates DM, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J, Hornik K, Hothorn T, Huber W, Iacus S, Irizarry R, Leisch F, Li C, Maechler M, Rossini AJ, Sawitzki G, Smith C, Smyth G, Tierney L, Yang JYH, Zhang J (2004). "**Bioconductor**: Open Software Development for Computational Biology and Bioinformatics." *Genome Biology*, **5**(10), R80. doi:10.1186/gb-2004-5-10-r80.

Genz A, Bretz F (2009). *Computation of Multivariate Normal and t Probabilities*. 1st edition. Springer-Verlag. doi:10.1007/978-3-642-01689-9.

Gogarten SM, Bhangale T, Conomos MP, Laurie CA, McHugh CP, Painter I, Zheng X, Crosslin DR, Levine D, Lumley T, Nelson SC, Rice K, Shen J, Swarnkar R, Weir BS, Laurie CC (2012). "**GWASTools**: An R/**Bioconductor** Package for Quality Control and Analysis of Genome-Wide Association Studies." *Bioinformatics*, **28**(24), 3329–3331. doi:10.1093/bioinformatics/bts610.

Gonzalez JR, Armengol L, Solé X, Guino E, Mercader JM, Estivill X, Moreno V (2007). "**SNPassoc**: An R Package to Perform Whole Genome Association Studies." *Bioinformatics*, **23**(5), 654–655. doi:10.1093/bioinformatics/btm025.

Hiersche M, Rühle F, Stoll M (2013). "**postgwas**: Advanced GWAS Interpretation in R." *PLoS ONE*, **8**(8), 1–10. doi:10.1371/journal.pone.0071775.

Hill WG, Robertson A (1968). "Linkage Diseqilibrium in Finite Populations." *Theoretical and Applied Genetics*, **38**(6), 226–231. doi:10.1007/bf01245622.

Hoffman GE, Logsdon BA, Mezey JG (2013). "PUMA: A Unified Framework for Penalized Multiple Regression Analysis of GWAS Data." *PLoS Computational Biology*, **9**(6), 1–19. doi:10.1371/journal.pcbi.1003101.

Howie BN, Donnelly P, Marchini J (2009). "A Flexible and Accurate Genotype Imputation Method for the Next Generation of Genome-Wide Association Studies." *PLoS Genetics*, **5**(6), e1000529. doi:10.1371/journal.pgen.1000529.

Huang H, Chanda P, Alonso A, Bader JS, Arking DE (2011). "Gene-Based Tests of Association." *PLoS Genetics*, **7**(7), e1002177. doi:10.1371/journal.pgen.1002177.

Jiang B, Zhang X, Zuo Y, Kang G (2011). "A Powerful Truncated Tail Strength Method for Testing Multiple Null Hypotheses in One Dataset." *Journal of Theoretical Biology*, **277**(1), 67–73. doi:10.1016/j.jtbi.2011.01.029.

Jombart T, Ahmed I (2011). "**Adegenet** 1.3-1: New Tools for the Analysis of Genome-Wide SNP Data." *Bioinformatics*, **27**(21), 3070–3071. doi:10.1093/bioinformatics/btr521.

Jorgenson E, Witte JS (2006). "A Gene-Centric Approach to Genome-Wide Association Studies." *Nature Review Genetics*, **7**(11), 885–891. doi:10.1038/nrg1962.

Karatzoglou A, Smola A, Hornik K, Zeileis A (2004). "**kernlab** – An S4 Package for Kernel Methods in R." *Journal of Statistical Software*, **11**(9), 1–20. doi:10.18637/jss.v011.i09.

Kwak IY, *et al.* (2016). ***aSPU**: Adaptive Sum of Powered Score Test*. R package version 1.41.

Larson NB, Jenkins GD, Larson MC, Vierkant RA, Sellers TA, Phelan CM, Schildkraut JM, Sutphen R, Pharoah PPD, Gayther SA, Wentzensen N, Goode EL, Fridley BL (2014). "Kernel Canonical Correlation Analysis for Assessing Gene-Gene Interactions and Application to Ovarian Cancer." *European Journal of Human Genetics*, **22**(1), 126–131. doi:10.1038/ejhg.2013.69.

Larson NB, Schaid DJ (2013). "A Kernel Regression Approach to Gene-Gene Interaction Detection for Case-Control Studies." *Genetic Epidemiology*, **37**(7), 695–703. doi:10.1002/gepi.21749.

Lê S, Josse J, Husson F (2008). "**FactoMineR**: A Package for Multivariate Analysis." *Journal of Statistical Software*, **25**(1), 1–18. doi:10.18637/jss.v025.i01.

Lee S, Zhao Z, Miropolsky L, Wu M (2020). ***SKAT**: SNP-Set (Sequence) Kernel Association Test*. R package version 2.0.0, URL https://CRAN.R-project.org/package=SKAT.

Li J, Huang D, Guo M, Liu X, Wang C, Teng Z, Zhang R, Jiang Y, Lv H, Wang L (2015). "A Gene-Based Information Gain Method for Detecting Gene-Gene Interactions in Case-Control Studies." *European Journal of Human Genetics*, **23**(11), 1566–1572. doi:10.1038/ejhg.2015.16.

Li J, Ji L (2005). "Adjusting Multiple Testing in Multilocus Analyses Using the Eigenvalues of a Correlation Matrix." *Heredity*, **95**(3), 221–227. doi:10.1038/sj.hdy.6800717.

Li J, Tang R, Biernacka J, de Andrade M (2009). "Identification of Gene-Gene Interaction Using Principal Components." *BMC Proceedings*, **3**(Suppl 7), S78. doi:10.1186/1753-6561-3-s7-s78.

Li MX, Gui HS, Kwan JSH, Sham PC (2011). "GATES: A Rapid and Powerful Gene-Based Association Test Using Extended Simes Procedure." *The American Journal of Human Genetics*, **88**(3), 283–293. doi:10.1016/j.ajhg.2011.01.019.

Lin C (2016). ***etma**: Epistasis Test in Meta-Analysis*. R package version 1.1-1, URL https://CRAN.R-project.org/package=etma.

Ma L, Clark AG, Keinan A (2013). "Gene-Based Testing of Interactions in Association Studies of Quantitative Traits." *PLoS Genetics*, **9**(2), e1003321. doi:10.1371/journal.pgen.1003321.

Matthews GJ, Foulkes AS (2015). "**MixMAP**: An R Package for Mixed Modeling of Meta-Analysis p Values in Genetic Association Studies." *Journal of Statistical Software*, **66**(3), 1–11. doi:10.18637/jss.v066.c03.

McHugh C, Larson J, Hackney J (2020). ***cpvSNP**: Gene Set Analysis Methods for SNP Association p-Values That Lie in Genes in Given Gene Sets*. doi:10.18129/B9.bioc.cpvSNP. R package version 1.20.0.

Moskvina V, Schmidt KM (2008). "On Multiple-Testing Correction in Genome-Wide Association Studies." *Genetic Epidemiology*, **32**(6), 567–573. doi:10.1002/gepi.20331.

Neale BM, Sham PC (2004). "The Future of Association Studies: Gene-Based Analysis and Replication." *The American Journal of Human Genetics*, **75**(3), 353–362. doi:10.1086/423901.

Nyholt DR (2004). "A Simple Correction for Multiple Testing for Single-Nucleotide Polymorphisms in Linkage Disequilibrium with Each Other." *The American Journal of Human Genetics*, **74**(4), 765–769. doi:10.1086/383251.

Peng Q, Zhao J, Xue F (2010). "A Gene-Based Method for Detecting Gene-Gene Co-Association in a Case-Control Association Study." *European Journal of Human Genetics*, **18**(5), 582–587. doi:10.1038/ejhg.2009.223.

Phillips PC (2008). "Epistasis – The Essential Role of Gene Interactions in the Structure and Evolution of Genetic Systems." *Nature Review Genetics*, **9**(11), 855–867. doi:10.1038/nrg2452.

Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, Maller J, Sklar P, de Bakker PIW, Daly MJ, Sham PC (2007). "**PLINK**: A Toolset for Whole-Genome Association and Population-Based Linkage Analysis." *American Journal of Human Genetics*, **81**, 559–575. doi:10.1086/519795.

Rajapakse I, Perlman MD, Martin PJ, Hansen JA, Kooperberg C (2012). "Multivariate Detection of Gene-Gene Interactions." *Genetic Epidemiology*, **36**(6), 622–630. doi:10.1002/gepi.21656.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Sanchez G, Trinchera L, Russolillo G (2017). ***plspm**: Tools for Partial Least Squares Path Modeling (PLS-PM)*. R package version 0.4.9, URL https://CRAN.R-project.org/src/contrib/Archive/plspm/.

Schwender H (2020). ***logicFS**: Identification of SNP Interactions*. doi:10.18129/B9.bioc.logicFS. R package version 2.8.0.

Solé X, Guinó E, Valls J, Iniesta R, Moreno V (2006). "**SNPStats**: A Web Tool for the Analysis of Association Studies." *Bioinformatics*, **22**(15), 1928–1929. doi:10.1093/bioinformatics/btl268.

Wan X, Yang C, Yang Q, Xue H, Fan X, Tang NLS, Yu W (2010). "BOOST: A Fast Approach to Detecting Gene-Gene Interactions in Genome-Wide Case-Control Studies." *The American Journal of Human Genetics*, **87**(3), 325–340. doi:10.1016/j.ajhg.2010.07.021.

Winham S (2012). ***MDR**: Detect Gene-Gene Interactions Using Multifactor Dimensionality Reduction*. R package version 1.2, URL https://CRAN.R-project.org/src/contrib/Archive/MDR/.

Wu MC, Kraft P, Epstein MP, Taylor DM, Chanock SJ, Hunter DJ, Lin X (2010). "Powerful SNP-Set Analysis for Case-Control Genome-Wide Association Studies." *American Journal of Human Genetics*, **86**(6), 929–942. doi:10.1016/j.ajhg.2010.05.002.

Yuan Z, Gao Q, He Y, Zhang X, Li F, Zhao J, Xue F (2012). "Detection for Gene-Gene Co-Association via Kernel Canonical Correlation Analysis." *BMC Genetics*, **13**(1), 83. doi:10.1186/1471-2156-13-83.

Zaykin DV, Zhivotovsky LA, Westfall PH, Weir BS (2002). "Truncated Product Method for Combining *P*-Values." *Genetic Epidemiology*, **22**(2), 170–185. doi:10.1002/gepi.0042.

Zhang X, Huang S, Zou F, Wang W (2010). "TEAM: Efficient Two-Locus Epistasis Tests in Human Genome-Wide Association Study." *Bioinformatics*, **26**(12), i217–i227. doi:10.1093/bioinformatics/btq186.

Zhang X, Yang X, Yuan Z, Liu Y, Li F, Peng B, Zhu D, Zhao J, Xue F (2013). "A PLSPM-Based Test Statistic for Detecting Gene-Gene Co-Association in Genome-Wide Association Study with Case-Control Design." *PLoS ONE*, **8**(4), e62129. doi:10.1371/journal.pone.0062129.

Affiliation:

Mathieu Emily
Department of Statistics
Agrocampus Ouest
35042 Rennes, France
E-mail: Mathieu.Emily@agrocampus-ouest.fr
URL: http://emily.perso.math.cnrs.fr/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
October 2020, Volume 95, Issue 12 *Submitted:* 2016-10-17
doi:10.18637/jss.v095.i12 *Accepted:* 2019-10-22