A tutorial for the Bioconductor package `hierGWAS`

Laura Buzdugan

April 30, 2018

Contents

1 Introduction

`hierGWAS` tests statistical significance in Genome Wide Association Studies (GWAS), using a joint model of all SNPs. This leads to a stronger interpretation of single, or groups of SNPs compared to the marginal approach. The method provides an asymptotic control of the Family Wise Error Rate (FWER), as well as an automatic, data-driven refinement of the SNP clusters to smaller groups or single markers [1].

The method can be used for case-control studies, as well as for continuous trait studies. Aditionally, one can control for one or more covariates.

1.1 Model

The two components of the model are the genotype X and the phenotype Y. Y is a vector of length n, where n represents the number of samples (e.g. subjects). Y_i, the phenotype corresponding to

person i, can encode either the disease status (0 or 1), or the continuous value of a trait. X is a matrix of size $n \times p$, where p is the number of SNPs to be analyzed. $X_{i,j}$ encodes the number of minor alleles of the jth SNP for person i, thus it can take values $0, 1, 2$.

The genotype-phenotype relationship is expressed using a generalized linear model [2]. If the phenotype is continuous, this translates to a linear model:

$$Y_i = \beta_0 + \sum_{j=1}^{p} \beta_j X_{i,j} + \epsilon_i \quad (i = 1, .., n)$$

where $\epsilon_i, .., \epsilon_n$ are independent and identically distributed error terms with expectation 0. If the phenotype is binary, representing case (=1) or control (=0) status, we use a logistic regression model:

$$\pi_i = P(Y_i = 1 | X_i, \beta) = \frac{exp(\eta_i)}{1 + exp(\eta_i)} \quad (i = 1, .., n)$$

$$\ln \frac{\pi_i}{1 - \pi_i} = \eta_i = \beta_0 + \sum_{j=1}^{p} \beta_j X_{i,j}$$

where π represents the probability of a subject having case status, given its SNPs X_i.

In both models β_0 represents the intercept and β_j are the (logistic) regression coefficients which quantify the association between the response and SNP j.

Our goal is to assess the statistical significance of single SNPs, or groups of correlated SNPs, with respect to the phenotype. This has to be made precise: we aim for p-values, adjusted for multiple testing, when testing the hypotheses: for single SNP j

$$H_{0,j} : \ \beta_j = 0 \text{ versus } H_{A,j} : \ \beta_j \neq 0, \tag{1}$$

or for a group $G \subseteq \{1, \ldots, p\}$ of SNPs

$$H_{0,G} : \ \beta_j = 0 \text{ for all } j \in G$$
$$\text{versus} \quad H_{A,G} : \ \text{at least for one } j \in G \text{ we have that } \beta_j \neq 0. \tag{2}$$

The obtained p-values are with respect to a regression model and hence, they share the interpretation with the regression parameters described above.

1.2 Workflow

The entire procedure for hierarchical inference is schematically summarized in Figure 1. Preprocessing refers to the steps taken to clean the data, by removing SNPs, samples. This should be performed before starting the analysis. The three stages of the procedure are described in detail in sections 3.1, 3.2 and 3.3.

2 Preparing Data

2.1 Data Formats

The two required variables in order to perform the analysis are the phenotype and genotype data. The genotype data, `x`, should be a matrix of size `n x p`, where `n` represents the number of samples (subjects), and `p` the number of SNPs. The SNPs must be coded numerically (0,1,2 copies of a specific allele). The phenotype data, `y` should be a vector of length `n`, and it should have either binary (0,1), or continuous valued elements.
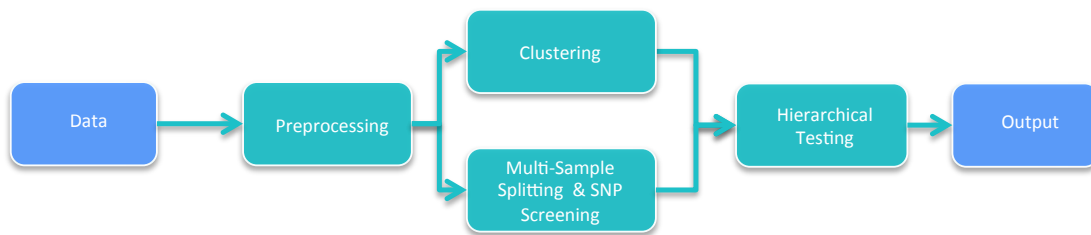
Figure 1: Flowchart of the method.

If one is working with `PLINK`, these two data structures can easily be created in `R` after the `PLINK` data file is read.

There are two optional input variables. `SNP_index`, a vector of length `p`, assigns a label to each SNP. These could represent chromosomes, or some other grouping of the SNPs. `covar`, either of vector of length `n`, or a matrix of size `n x c`, stores the covariates one wishes to control for.

2.2 Missingness

In general, even after preprocessing, SNP data contains missing values. While this is not an issue in a marginal analysis, where missing data are omitted, multivariate methods such as `lm`, `glm`, do not allow for data with missingness.

Thus, one must impute these missing values before starting the analysis. There are several different methods to do this, either in `R`, after the data is imported, or before. If done inside `R`, one can choose from packages such as `mice`, `mi` or `missforest`, which perform multiple imputation.

If one wishes to use methods tailored for GWAS data, software such as `SHAPEIT` [3] can be employed. Although the goal of `SHAPEIT` is to impute additional SNPs that were not genotyped, it will also impute the missing values for the genotyped SNPs. This is done in the process of phasing the chromosomes, when missing values are already imputed.

2.3 Toy Data

Given the usual size of GWAS data, performing the analysis on a real dataset with $> 10^5$ SNPs is computationally expensive. For this reason, in order to illustrate the method, we generated a small toy dataset using `PLINK`.

The data contains $n = 500$ samples (250 cases, 250 controls) and $p = 1000$ SNPs. Out of these 1000 SNPs, 990 have no association with the phenotype, while the remaining 10 have a population odds ratio of 2. The SNPs were binned into different allele frequency ranges, to create a more realistic example.

The SNPs are separated into two chromosomes: the first half of the SNPs are from chromosome 1, while the second half from chromosome 2. Finally, there are two control variables: sex (0 for males, 1 for females) and age (between 18 and 65).

The data structure, entitled simGWAS, has the following components:

- SNP.1: The first SNP, of dimension 500×1. Each row represents a subject.

- ...

- SNP.1000: The last SNP, of dimension 500×1. Each row represents a subject.

- y: The response vector. It can be continuous or discrete.

- sex: The first covariate, represeting the sex of the subjects: 0 for men and 1 for women.

- age: The second covariate, represeting the age of the subjects.

3   Data Analysis

The `hierGWAS` R-package contains the following functions:

Function	Description
cluster.snp	clusters the SNP hierarchically
multisplit	performs the multiple sample splitting and LASSO regression
test.hierarchy	hierarchically tests the SNPs
compute.r2	computes the r^2 for a group of SNPs

The first three functions perform the clustering, regression and testing. The outputs of `cluster.snp` and `multisplit` are required as inputs for `test.hierarchy`, so these two functions need to be executed before the third one. On the other hand, `cluster.snp` and `multisplit` are independent, so they can be run in any order. In order to speed up the computations, one should consider running them in parallel if the resoures allow it. The last function, `compute.r2`, allows the user to calculate the variance explained by any group of SNPs of arbitrary size.

```
> library(hierGWAS)
> data(simGWAS)
> sim.geno <- data.matrix(simGWAS[,1:1000])
> sim.pheno <- simGWAS$y
> sim.covar <- cbind(simGWAS$sex,simGWAS$age)
```

In the following we will describe each of the four functions, and show how they should be used.

3.1   Hierarchical Clustering of SNPs

The first step is to hierarchically cluster the SNPs. `cluster.snp` uses the `hclust` function from R, but provides a different distance measure. Hierarchical clustering is a computationally intensive procedure, and for large datasets it becomes unfeasible. For this reason, one can divide the SNPs into non-overlapping sets, and cluster each set separately. One natural division is the grouping of SNPs into distinct chromosomes, but the user is free to define alternative divisions.

In the case of our toy data, the SNPs belong to either chromosome 1 or 2, so we cluster them into two distinct hierarchies, by specifying the indices of the SNPs which come from either chromosome 1 or 2. Note that in this case, due to the small size of the data, it would be equally feasible to cluster all the SNPs into a single tree.

```
> # cluster SNPs in chromosome 1
> SNPindex.chrom1 <- seq(1,500)
> dendr.chrom1 <- cluster.snp(sim.geno,SNP_index = SNPindex.chrom1)
> # cluster SNPs in chromosome 2
> SNPindex.chrom2 <- seq(501,1000)
> dendr.chrom2 <- cluster.snp(sim.geno,SNP_index = SNPindex.chrom2)
```

The inputs for `cluster.snp` are `x`, the genotype matrix, respectively `SNPindex.chrom1` and `SNPindex.chrom2`, the indices of SNPs in chromosomes 1 and 2. `x` was used to compute the dissimilarity measure between the SNPs.

The default dissimilarity measure between two SNPs is `1 - LD` (Linkage Disequilibrium), where `LD` is defined as the square of the Pearson correlation coefficient. If the user wishes to use a different dissimilarity measure, this will then be the sole input to `cluster.snp`. For example, if we were to use a different measure, the input would have been only the dissimilarity matrix. Thus, for our toy data, we would have had to provide either two 500×500 matrices, one per chromosome, and execute the function separately for each chromosome, or give one 1000×1000 dissimilarity matrix for the entire dataset.

Finally, the default agglomeration method is average linkage, however the user can choose between multiple methods implemented by `hclust`.

For large datasets, one can speed up the analysis by running the clustering of separate chromosomes in parallel. This can be done using the `parallel R` package.

3.2 Multiple Sample Splitting and LASSO Regression

The second step, which can be run independently from the first one, is to do repeated LASSO regressions on random sample splits. This part of the analysis is implemented in the `multisplit` function. This function takes as input the genotype and phenotype data, as well as one or more covariates.

The procedure is as follows:

1. Randomly split the sample into two equal parts: I_1 and I_2

2. Do LASSO selection on the data from I_1. Save the $n/6$ selected SNPs.

3. Repeat steps 1-2 B times. The default value for B is 50, but a different number of random splits can be specified.

The output of this function is a list with two components:

- out.sample: a matrix of size $B \times [n/2]$ containing the second subsample (I_2) for each split.

- sel.coeff: a matrix of size $B \times [n/6]$ containing the selected variables in each split.

For our example, the code is the following:

```
> res.multisplit <- multisplit(sim.geno,sim.pheno,covar = sim.covar)
> # the matrix of selected coefficients for each sample split
> show(res.multisplit$sel.coeff[1:10,1:10])

         [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
  [1,]   991  989  218  546  990  150  830  846  998   252
  [2,]   992  990  991  989  533  996  997  321  995   471
  [3,]   995  991  990  989  616  996  321  323  553   992
  [4,]   995  991  997  990  992  843  989  492  588   996
  [5,]   995  989  992  996  218  990  655  730  991   827
  [6,]   990  995  178  989  996  997  991  344  798   405
  [7,]   995  990  991  989  992  998  299  472  665   997
  [8,]   991  995  997  990  992  989  731  218  323   996
  [9,]   989  992  990  991  178  906  998  997  996   598
 [10,]   991  995  998  897  989  882  313  334  602   834
```

```
> # the samples which will be used for testing
> show(res.multisplit$out.sample[1:10,1:10])

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    4    7    9   10   11   13   14   15    16
 [2,]    1    3    4    7    8   10   11   12   14    18
 [3,]    1    2    4    5    6    9   10   11   14    15
 [4,]    1    2    4    7    8   11   13   14   16    17
 [5,]    1    2    6    7   10   11   13   16   17    18
 [6,]    4    6    7   10   11   13   14   15   16    19
 [7,]    2    3    5    6    7   10   11   12   13    15
 [8,]    3    5    6    7    8    9   11   12   21    23
 [9,]    4    5    9   10   11   12   19   20   21    22
[10,]    2    3    4    5    7   19   21   22   24    25
```

3.3   Hierarchical Testing of SNPs

The last step of the procedure is the hierarchical testing of SNPs. We describe the steps for the toy data:

1. Test the global hypothesis $H_{0,G_{\mathrm{global}}}$ where $G_{\mathrm{global}} = \{1, \ldots, p\}$: that is, we test whether all SNPs have corresponding (generalized) regression coefficients equal to zero or alternatively, whether there is at least one SNP which has a non-zero regression coefficient. If we can reject this global hypothesis, we go to the next step.

2. Test the hypotheses $H_{0,G_1}, \ldots, H_{0,G_2}$ where G_k contains all the SNPs from chromosome k. For those chromosomes k where H_{0,G_k} can be rejected, we go to the next step.

3. Test hierarchically the groups G which correspond to chromosomes k where H_{0,G_k} is rejected: first consider the largest groups and then proceed hierarchically (down the cluster tree) to smaller groups until a hypothesis $H_{0,G}$ can not be rejected anymore or the level of single SNPs is reached.

4. The output is a collection of groups $G_{\mathrm{final},1}, \ldots, G_{\mathrm{final},m}$ where $H_{0,G_{\mathrm{final},k}}$ is rejected ($k = 1, \ldots, m$) and all subgroups of $G_{\mathrm{final},k}$ ($k = 1, \ldots, m$) downwards in the cluster tree are not significant anymore.

With this procedure, one needs to do the hypothesis tests in such a hierarchical procedure at certain levels to guarantee that the familywise error, i.e., the probability for at least one false rejection of the hypotheses among the multiple tests, is smaller or equal to α for some pre-spcified $0 < \alpha < 1$, e.g., $\alpha = 0.05$.

For our dataset, the `test.hierarchy` function takes as input the following variables:

- x: the genotype matrix

- y: the phenotype vector

- dendr: the hierarchical tree (of one of the chromosomes)

- res.multisplit: the output of the LASSO regression

- SNP_index: the indices of SNPs (in one of the chromosomes)

- global.test: specifies wether the global null hypothesis should be tested

- verbose: reports information on progress

We then run the function for both chromosomes 1 and 2.

```
> result.chrom1 <- test.hierarchy(sim.geno, sim.pheno, dendr.chrom1, res.multisplit,
+                                  covar = sim.covar, SNP_index = SNPindex.chrom1)
> result.chrom2 <- test.hierarchy(sim.geno, sim.pheno, dendr.chrom1, res.multisplit,
+                                  covar = sim.covar, SNP_index = SNPindex.chrom2,
+                                  global.test = FALSE, verbose = FALSE)
> show(result.chrom2)

[[1]]
[[1]]$label
[1] 992

[[1]]$pval
[1] 0.01537805


[[2]]
[[2]]$label
[1] 991

[[2]]$pval
[1] 0.00155887


[[3]]
[[3]]$label
[1] 998

[[3]]$pval
[1] 0.01922347


[[4]]
[[4]]$label
[1] 997

[[4]]$pval
[1] 0.02523474
```

The ouptput of `test.hierarchy` is a list containing groups or individually significant SNPs, as well as their p-values.

Based on the indices of the significant SNPs, one can go back to the original data and retrieve the SNP rsIDs.

Due to the fact that the procedure begins with testing the globall null $H_{0,G_{\text{global}}}$, if we subdivide the SNPs into chromosomes or other groups, we will end up testing the global null for each division. In order to save time, once the global null has been rejected (e.g. for chromosome 1), when testing chromosome 2, the `test.global` parameter should be set to `FALSE`, to avoid retesting the null.

The final parameter of the function, `verbose`, offers the user updates about the progress of the algorithm. By default this parameter is set to `TRUE`, but the messages can be suppressed by setting it to `FALSE`.

3.4 Computation of explained variance

One has the option of calculating the variance explained by any group of SNPs, using the `compute.r2` function. The inputs are:

- x: the genotype matrix

- y: the phenotype vector

- res.multisplit: the output of the LASSO regression

- covar: the matrix of covariates

- SNP_index: the indices of SNPs whose r^2 will be computed

The r^2 of a group is computed in the following way:

1. Intersect the group indices with those of the selected coefficients in split b

2. Compute the r^2 of the model with SNPs from point 1.

3. Repeat steps 1-2 B times. The final r^2 is the mean of the B r^2 values.

For our toy data, given that we know a-priori that the last 10 SNPs are the ones that increase the risk of disease, we chose to compute the variance explained jointly by them. This is done using the following command:

```
> SNP_index <- seq(991,1000)
> res.r2 <- compute.r2(sim.geno, sim.pheno, res.multisplit, covar = sim.covar, SNP_index = SNP_ind
> show(res.r2)

[1] 0.2498904
```

4 Session Info

```
> sessionInfo()

R version 3.5.0 (2018-04-23)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.4 LTS

Matrix products: default
BLAS: /home/biocbuild/bbs-3.7-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.7-bioc/R/lib/libRlapack.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
```

```
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] hierGWAS_1.10.0

loaded via a namespace (and not attached):
 [1] compiler_3.5.0    Matrix_1.2-14    tools_3.5.0      codetools_0.2-15
 [5] grid_3.5.0        iterators_1.0.9  foreach_1.4.4    fmsb_0.6.3
 [9] fastcluster_1.1.24 glmnet_2.0-16   lattice_0.20-35
```

References

[1] L. Buzdugan et al. *Assessing statistical significance in predictive genome-wide association studies.* (unpublished).

[2] P. McCullagh and J.A. Nelder. *Generalized Linear Models.* Chapman & Hall, London, 1989.

[3] O. Delaneau et al. *Improved whole-chromosome phasing for disease and population genetic studies.* Nature Methods, 10(1):5-6, 2013.