The **GWAS Lasso** Package

Version 0.20

NatingWang

1. Introduction

The GWAS Lasso package is developed to identify significant SNPs that control phenotypic variation and estimate their additive and dominant genetic effects based on the Bayesian Lasso Model and Bayesian Group Lasso model. These two statistical models are cornerstone for identifying the relation between genes and traits in this GWAS Lasso package. The first Bayesian Lasso named ***BLS*** model in this package can detect the associations using measurements at a single time point, while the second group Lasso named ***GLS*** model is capable of consuming the longitudinal phenotypes. This guide gives a brief instruction on how to perform the tasks of SNP detection by the GWAS Lasso package. The outline of this guide is as follows:

Section 2: Installation

Section 3: Data Format

Section 4: Statistical Models

Section 5: Summary

Section 6: Plot

Section7: Parallel Computing

We refer to Li et al. (2011) and Li et al (2015) for the theoretical foundation of this package. If you benefit from this software, please cite the following papers in your work:

1. Li, J., Das, K., Fu, G., Li, R., & Wu, R. (2011). The Bayesian Lasso for genome-wide association studies. Bioinformatics, 27(4), 516-523.

2. Li. J., Wang, Z., Li, R., Wu, R. (2015). Bayesian Group Lasso for nonparametric varying-coefficient models with application to functional genome-wide association studies. The Annals of Applied Statistics. 9(1).

2. Installation

The GWAS Lasso package depends on the ***snpStats***, ***nlme*** and ***snowfall*** package (specifically, the ***snpStats*** and ***nlme*** package are required, but the ***snowfall*** package is included for parallel computation and is optional), so these packages should be installed firstly. To install GWAS Lasso, download the package file and type the appropriate command in Linux command window or R console window.

1) For Linux command Window

```
$ R CMD INSTALL gwas.lasso_0.20.tar.gz
```

2) For R console window

```
>install.packages("gwas.lasso_0.20.tar.gz")
```

Before the package is used in R, the package importation is necessary by the following R command:

```
>library(gwas.lasso)
```

3. Data format

The GWAS Lasso package can detect the joint effects of multiple significant SNPs through the analysis of the data files with appropriately formatted genotype and phenotype information. There are two genotypic data formats and two phenotypic data formats used in this package.

3.1. Phenotypic data

For two statistical models, the phenotypic data take the CSV (Comma-separated values) file containing individual identification numbers, covariates as well as response values.

3.1.1. One single measure for the BLS model

The example below demonstrates a phenotype file contains ID, two covariates (X_1, X_2) and one phenotype value measured at a single time point (Y).

```
ID,X_1,X_2,Y
1,0,0.663,33.72
2,1,0.728,36.78
3,0,NA,38.92
…
```

The phenotype file should be filled with numerical values and missing values are encoded as NA. There are limited data checks in the current version, and the package requires the individual IDs in the phenotype file to be consistent with those in the genotype file. In this example, one row represents one subject. The covariate columns are optional but the phenotype column is required to call the functions of BLS model.

3.1.2. Longitudinal data for the GLS model

The GLS model, which can estimate time varying curves for additive and dominant effects of each significant SNP, requires the longitudinal traits as phenotypic data. The example below lists one ID column (ID), two time-invariant covariate columns (X_1, X_2), measurement time columns (Z_1, Z_2, Z_3, Z_4, Z_5, …) and phenotype columns (Y_1, Y_2, Y_3, Y_4, Y_5, …). In this example, X_* columns stand for covariate values, Z_* columns stand for measurement times and Y_* columns stand for longitudinal phenotypic values.

```
ID,X_1,X_2,Z_1,Z_2,Z_3,Z_4,Z_5,…, Y_1,Y_2,Y_3,Y_4,Y_5,…
1, 1,-0.946,40.000,42.000,43.000,48.000,50.000,…,18.853,14.289,11.529,15.920,22.203,…
2, 3, -0.846,41.000,40.000,38.000,39.000,NA,…,17.853,12.289,13.529,15.920,NA,…
…
```

3.2. Genotypic data

Genotypic data are supposed to be huge or mess if hundreds of thousands SNPs are stored. In general, PLINK is a very common and powerful tool to compress, convert and analyze these big data. This package not only employs PLINK to pack genotype data, but also allows a user-defined, simple format to store some small genotype data which are produced by small experiments or outputted by other package.

3.2.1. PLINK format

Given the advantages of storage space and loading time, the binary PLINK files are used by default in this package. If the binary data file is not readily available, the following command can convert the common PED and MAP paired files into the binary group files, which include one binary file (*.bed) and two plain text files (*.bim and *.fam) that can be viewed with a standard text editor.

```
$ plink --file mydata --out mydata --make-bed
```

1) *bed* file

The *bed* file is a compressed binary file containing genotype information. If you try to view it, you will only see lots of strange characters on the screen.

2) *bim* file

The *bim* file is an extended MAP file where each line of this file describes a single individual and it must contain exactly 4 columns: chromosome, SNP identifier, genetic distance and base-pair position (bp units). The following two extra columns are allele names.

```
0    ss66369915    0    0    G    A
0    ss66112992    0    0    G    A
…
```

3) *fam* file

Phenotypic information are stored in the *fam* file where one row represents one subject and the first six columns are mandatory: Family ID, Individual ID, Paternal ID, Maternal ID, Sex (1=male; 2=female; other=unknown) and phenotype. However, the phenotype defined here is ***not*** used in this package, as a separate phenotype data will be supplied.

```
957 2274 13631 2615 2 30.6367470222222
137 2349 0 0 2 34.4484237154545
…
```

3.2.1 Simple format

In addition to the PLINK format, a user-defined format named simple format is designed to store small amount of SNPs for users who do not use PLINK. The genotypic data are stored in the CSV format, where each line describes a single SNP and must start with 2 columns of chromosome information (chromosome number and SNP position). Three genotypes (*aa*=0, *Aa*=1, *AA*=2) and missing data (coded as -1 or *NA*) are valid SNP values.

```
CHR,POS,Sub1,Sub2,Sub3,Sub4,Sub5, …
0,1, 2, 0, 1, 1, 1, …
0,2, 2, 1, 1, 0, 0, …
…
```

4. Statistical model

This package implements two statistical models: the BLS model and GLS model. The difference between two models is whether the longitudinal phenotype can be handled. More specifically, phenotype measured at a single time point is analyzed in the BLS model, while longitudinal data that discover the dynamic patterns of the traits are expected in GLS model.

4.1 The BLS model

In BLS model, a number of important covariates, which are either discrete or continuous, along with additive and dominant effects are integrated into one statistical framework where effects are estimated jointly. The response value y_i measured at one time point for subject *i* is dissected in equation (1).

$$y_i = \mu + X_i^T\alpha + Z_i^T\beta + \xi_i^T a + \zeta_i^T d + \epsilon_i, \qquad i = 1, \ldots, n, \qquad (1)$$

where μ is the overall mean, X_i is the vector of discrete covariates, a is the vector of regression coefficients for discrete covariates, Z_i is the vector of continuous covariates, β is the vector of regression coefficients for continuous covariates, $a = (a_1,\ldots, a_p)^T$ and $d = (d_1,\ldots, d_p)^T$ are the p-dimensional vectors of the additive and dominant effects of SNPs, respectively, ξ_i and ζ_i are the indicator vectors of the additive and dominant effects of SNPs, and ϵ_i isthe residual error assumed to follow a $N(0; \sigma^2)$ distribution. The *j*-th elements of ξ_i and ζ_i are defined as

$$\xi_{ij} = \begin{cases} 1, \text{if the genotype of SNP j is AA} \\ 0, \text{if the genotype of SNP j is Aa} \\ -1 \text{ if the genotype of SNP is aa} \end{cases}$$

$$\zeta_{ij} = \begin{cases} 1, \text{if the genotype of SNP j is Aa} \\ 0, \text{if the genotype of SNP is AA or aa} \end{cases}$$

4.2 The GLS model

The GLS model employs the Legendre Polynomials to fit the time-varying genetic effects of each subject, where phenotypic data are measured repeatedly over time and are stored in the longitudinal data. Let y_i be the T_i-dimensional vector of measurements on subject *i* where $t_i = (t_{i1}, \ldots, t_{iTi})^T$ is the corresponding vector of measurement time points after standardization. Thus, at time point t_{ij},

$$y_i(t_{il}) = \mu(t_{il}) + \alpha(t_{il})^T X_i + a(t_{il})^T \xi_i + d(t_{il})^T \zeta_i + e_i(t_{il}) \quad i = 1, \dots, n; l = 1, \dots, T_i$$

where $\mu(t_{il})$ is the vector of the overall mean, $\alpha(t_{il})$ is the a vector of covariate effects, X_i be the observed covariate vector, $a_j(t_{il})$ is additive effect of j-th SNP, $d_j(t_{il})$ is dominant effect of j-th SNP, and $e_i(t_{il})$ is the residual error assumed to follow the $N(0; \sigma^2(t_{il}))$ distribution. The j-th elements of ξ_i and ζ_i are defined as above.

In the GLS model, the effects of covariates and SNPs are assumed to be a time-varying curve modeled by the 4th-order Legendre Polynomials.

$$L(t_{il}) = r_0 + r_1 * t_{il} + r_2 * \frac{1}{2}(3t_{il}^2 - 1) + r_3 * \frac{1}{2}(5t_{il}^3 - 3t_{il})$$

By approximating time-varying effects using Legendre polynomials, the expansion coefficients can be solved through regression, and the effect of k-th covariate can be approximated by a Legendre polynomial of order 4.

5. Work flow

5.1. Preparing data

The package have two types of genotypic data: PLINK format or simple CSV file described in 3.2.1. Whichever format is used, the *imputation* and *quality control* are supposedly finished according to the experiment requirement. The package ***does not provide any quality control functions***. For the imputation function, the package only can impute missing SNPs based on the genotype frequency.

Although PLINK has simple phenotypic data, the package use the user-defined CSV files to store the phenotypic data. The definition has been elaborated in 3.1.

The simulation function is provided for users who would like to try this package without any real data. The BLS model and GLS model have respective simulation functions as follows.

```
bls.simulate("bls.phe.csv", "bls.gen.csv");
gls.simulate ("gls.phe.csv", "gls.gen.csv");
```

5.2 Starting computation

Each statistical model has three functions to analyze PLINK data, simple SNP data and SNP matrix data. In total, there are six functions available.

```
r.bls <- bls.simple("bls.phe.csv", "bls.gen.csv", Y.name="Y", covar.names=c("X_1","X_2"));
r.bls <- bls.plink (file.phe.long, file.plink.bed, file.plink.bim, file.plink.fam, Y.name="Y", covar.names=c());
r.bls <- bls.snpmat(tb.phe, tb.snp, Y.name="Y", covar.names=c("X_1","X_2"));

r.gls <- gls.simple("gls.phe.csv", "gls.gen.csv",Y.prefix="Y",Z.prefix="Z", covar.names=c("X_1","X_2"));
r.gls <- gls.plink(file.phe.long, file.plink.bed, file.plink.bim, file.plink.fam, Y.prefix="Y", Z.prefix="Z",
covar.names=c("X"));
```

```
r.gls <- gls.snpmat(tb.phe, tb.snp, Y.prefix="Y", Z.prefix="Z", covar.names=c("X_1","X_2"))
```

The function ***bls.simple*** and ***gls.simple*** are designed to load files encoded by the Simple format, ***bls.plink*** and ***gls.plink*** are intended to load PLINK files, but ***bls.snpmat*** and ***gls.snpmat*** are aimed to loading the genotype matrix rather than a data file.

Although parameters for data source are different, the control parameters are same for the overall six functions. Four types of control parameters allow the users to supply the following information.

1) Response variables (*Y.name*) and covariate names (*covar.name*)
2) If the filter implemented by fGWAS is used to pre-select significant SNPs?
3) If either additive or dominant effect is estimated?
4) If only variable selection procedure is applied to data analysis?

The whole computation task will involve several procedures, including data loading, SNP filter by fGWAS model if applicable, variable selection procedure and the final refit procedure. The following example shows how to do a computation using variable selection procedure only.

```
>   ret1   <-   bls.plink(   file.phe.long,   file.plink.bed,   file.plink.bim,
file.plink.fam, Y.name="Y", covar.names=c(),    fgwas.filter = T,  refit = F ,
options=list(nParallel.cpu=7));
[ BLASSO PLINK ] Procedure.
Checking the parameters ......
* Phenotypic Data File =  /work/bmi-pheno-age-mean.csv
* PLINK BED File =  /work/FHS-bmi-v1-chr2.bed
* PLINK BIM File =  /work/FHS-bmi-v1-chr2.bim
* PLINK FAM File =  /work/FHS-bmi-v1-chr2.fam
* Response Variable = Y
* Covariate Columns =
* fGWAS Filter Used = Yes
* Additive Effects Used = Yes
* Dominant Effects Used = Yes
* Refit Procedure = No
Checking the optional items......
* Parallel Computing:  Yes, 7 CPU(s)
* Piecewise Ratio:  2
* Threshold of fGWAS filter:  0.05
* Iteration of  Markov chain:  2000
* fBurnInRound:  0.3
* fRhoTuning:  0.095
* fQval.add:  0.05
* fQval.dom:  0.09
* Debug Output:  No
SNP Filtering by fGWAS method......
* SNP Count = 29295
* Sample Count = 819
* p-value Threshold = 0.05
  Calculated SNP Range = 1 20000
* H0 = Y ~ 1
* H1 = Y ~ 1 + as.factor(SNP)
  Starting parallel computing, snowfall/snow......
R Version:  R version 3.1.2 (2014-10-31)
  Stopping parallel computing......
  SNPs with p-value <= 0.05 : 1634
* 1634 SNPs( 5.58 %) are left after fGWAS filtering.
Genetic Effect Analysis by BLASSO/GLASSO method......
* Final LASSO calling.
Wrapping the results ......
```

5.3. Summarizing results

In the R console, ***summary*** is a generic function used to show the summary information of objects exported from the **BLS** model or the **GLS** model. It may include:

1) Significant SNPs estimated by the fGWAS method

2) Covariates coefficients estimated in the variable selection procedure

3) Pre-selected SNPs and effects estimated in the variable selection procedure

4) Covariates coefficients estimated in the refit procedure

5) Significant SNPs and effects estimated in the final refit procedure

```
summary( r.bls );
summary(r.gls);
```

5.4. Plotting figures

The function *plot* can output three types of PDF figure**,** including:

1) The Manhattan figure exported by the fGWAS method (*.fgwas.pdf)

2) The genetic effects of all SNPs estimated by the variable select procedure (*. varsel.pdf.)

3) The genetic effects of significant SNPs estimated by the refit procedure (*.refit.pdf).

The **BLS** result object or **GLS** result object are required to call the function *plot*.

```
plot (r.bls, fig.prefix="bls-ret");
plot (r.gls, fig.prefix="gls-ret");
```

The Manhattan figure gives –log10 (p-values) for each SNP, **from which the SNP with –log10 p-values greater than the threshold value specified in the control parameters will be selected to variable selection.** The figures of genetic effects are different depending on which model are used. The BLS model will output heritability sub-graph, but the GLS model outputs the time-varying additive and dominant effects for each significant SNP.

6. Internal Details

6.1 Simulation

Simulation is a good way to understand the functions and learn how to use them. The simulation in this package uses the pre-defined parameters to create a data object. This data object has the same structure as real data. All pre-defined parameters can be customized. The following two tables show the common part of simulation parameters.

Table1： Simulation Parameters of **BLS** Model

Items	**Description**
phe.out	String, the name of the output phenotypic data file
snp.out	String, the name of the output genotypic data file
simu_n	Integer, the size of samples
simu_p	Integer, the number of SNPs

simu_snp_rho	Float, correlation coefficient between two adjacent SNPs
simu_rho	Float, correlation coefficient between two adjacent time points
simu_sigma2	Float, the residual error assumed to follow a N(0,σ2(til)) distribution
simu_mu	Float, the overall mean of phenotypic data
simu_a_pos	Vector, positions of the significant SNPs with additive effects
simu_a_effect	Vector, additive effects of significant SNPs.
simu_d_pos	Vector, positions of the significant SNPs with dominant effects
simu_d_effect	Vector, dominant effects of significant SNPs.
simu_cov_range	Vector, the range of covariates
simu_t_range	Vector, the range of time points

Table2：Simulation Parameters of **GLS** Model

items	**Description**
file.phe.out	String, the name of the output phenotypic data file
file.snp.out	String, the name of the output genotypic data file
simu_n	Integer, the size of samples
simu_p	Integer, the number of SNPs
simu_snp_rho	Float, correlation coefficient between two adjacent SNPs
simu_rho	Float, correlation coefficient between two adjacent time points
simu_sigma2	Float, the residual error assumed to follow a N(0,σ2(til)) distribution
simu_mu	Vector, the overall mean of phenotypic data
simu_add_effect	Matrix, additive effects of significant SNPs where the first column represents positions of these SNPs
simu_dom_effect	Matrix, dominant effects of significant SNPs where the first column represents positions of these SNPs
simu_covar_effect	Matrix, effects of covariates
simu_covar_range	Vector, the range of covariates
simu_z_range	Numeric, the number of measurements
simu_z_count	Numeric, the number of measurements

bls.simulate and ***gls.simulate*** function can create two simulation data files: the phenotypic data file and the genotypic data file. The following codes show how to create simulation data and how to change some parameters by R command.

```
bls.simulate("bls.simple.phe", "bls.simple.snp");
bls.simulate("bls.simple.phe", "bls.simple.snp", simu_sigma2=4, simu_mu=26);
```

These above examples show how to use the BLS model. The first command generates simulation data by default parameter values, and the second one creates data using some customized parameters. On the other hand, the following example shows how to generate data for the GLS model.

```
a_effect <- array(c(1,2, runif(8,-4,5)), dim=c(2,5))
d_effect <- array(c(1,2, runif(8,-4,5)), dim=c(2,5))
```

```
gls.simulate ("gls.phe.mat.csv", "gls.snp.mat.csv", simu_n=300, simu_p=3000, simu_snp_rho=0.4,
simu_rho=0.1, simu_add_effect=a_effect, simu_dom_effect=d_effect,);
```

6.2. The SNP filter by fGWAS method

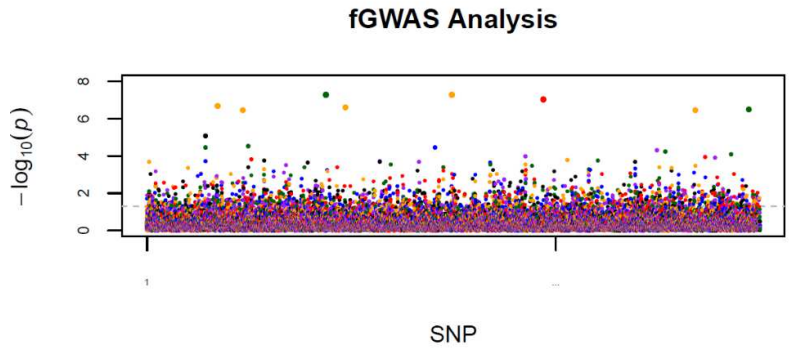
If the SNP number is extremely large, formulating Bayesian Group Lasso to select significant SNPs may lower the selection power and increase the false positive rate, so first filtering out less significant SNPs could avoid the problem.

The filter results can be viewed by two methods: summarizing the result object or plotting the associated Manhattan figure, or

1) Significant SNPs estimated by fGWAS method;

2) A figure gives –log10 p-values for each SNP, from which the SNP with –log10 p-values greater than the threshold value set before will be selected.

```
--- Significant SNPs Estimate by fGWAS method:
Top 25 SNPs:
SNP.ID CHR  POS    L.Ratio         pv
G0-SNP1271   1271   0 1271 117.167049 0.000000000
G0-SNP6728   6728   0 6728 112.728633 0.000000000
G0-SNP6883   6883   0 6883 260.680776 0.000000000
G0-SNP6899   6899   0 6899 102.708899 0.000000000
G0-SNP7897   7897   0 7897 151.950033 0.000000000
G0-SNP1683   1683   0 1683  13.695838 0.001061663
…….
```



6.3. Variable selection

This step formulates Bayesian Group Lasso to select a subset of significant SNPs by selecting Legendre coefficients that are not identically zero, at the same time, to estimate additive and dominant effects of significant SNPs. The unknown parameters could be estimated by using the MCMC algorithm.

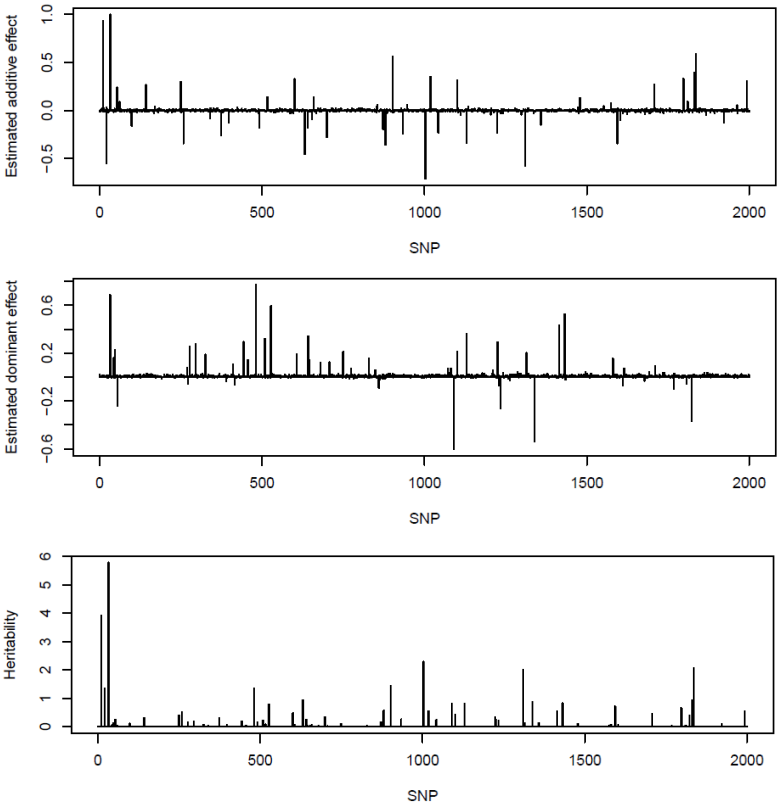
The results of variable selection include the following:

1) Variable Selection Result

2) A figure showing addictive effects and dominant effects of selected SNPs

3) A figure showing heritability of selected SNPs.

```
--- Covariate Estimate in Refit Procedure:
           Mode     L1     L2    L3     L4
Intercept 194.903 13.079 -3.153 1.928 -3.191
X_1       10.291  2.629 -1.512 0.894  0.541
--- Refit Result
chr  pos add.sig   add.mode     add.mu1     add.mu2     add.mu3     add.mu4
1   0 6728      2  0.3253068 -0.05632611 -0.13656087 -0.34259138 -0.45600212
2   0 6883      3  9.0963385  1.39688597 -2.43336304  1.10625151  0.05672361
3   0 6899      4  5.3871507  1.03125835  0.91000256 -1.73414760  0.69877344
4   0 7897      4 23.2958857  1.27028639 -0.32635179  0.95085732 -4.54660567
5   0 1271      0  0.0000000  0.04612361  0.03978521 -0.08060622  0.08770705
dom.sig  dom.mode    dom.mu1    dom.mu2    dom.mu3    dom.mu4
1       4  9.752713 1.33940947  0.9981518  2.0343131  1.6804636
2       4 33.455043 1.40182206 -2.0304856  5.0462396  1.3793230
3       2  0.114697 0.16038721  0.2982834 -0.1270506 -0.2547293
4       0  0.000000 0.09405818  0.2397407 -0.3175207 -0.2058939
5       4 13.728352 1.23633519 -1.1626640  2.7197024 -1.8577562
```



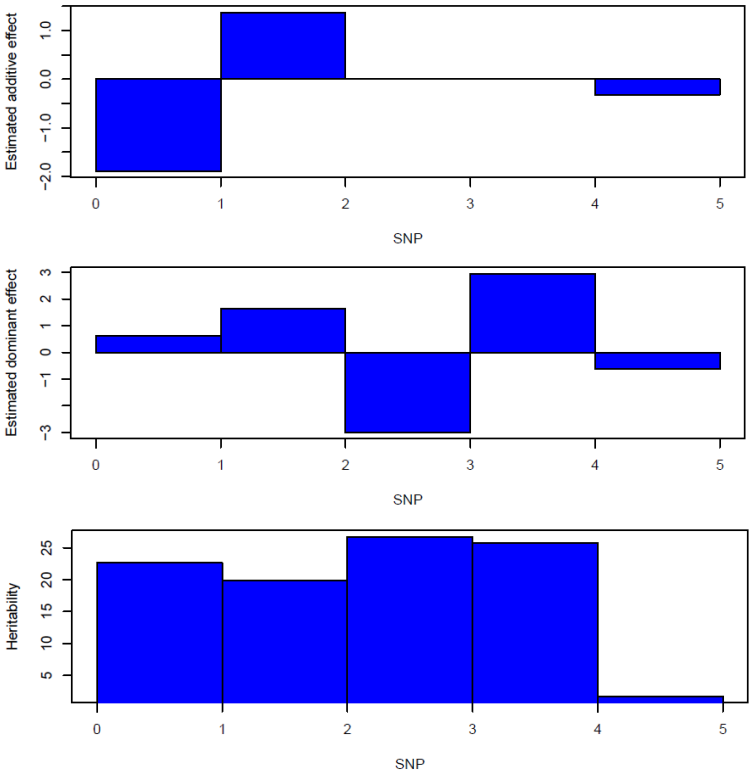
6.4. Refit

To ameliorate the bias of the parameter estimates introduced by Lasso penalties, it is necessary to refit the fGWAS model after variable selection, where only selected SNPs are included in the final model.

The results of result procedure include the following:

1) Significant SNPs and covariate estimation
2) A figure showing addictive effects and dominant effects of significant SNPs.
3) A figure showing heritability of significant SNPs.

```
--- Covariate Estimate in Refit Procedure:
           Mode     L1     L2    L3     L4
Intercept 194.903 13.079 -3.153 1.928 -3.191
X_1        10.291  2.629 -1.512 0.894  0.541
--- Refit Result
chr  pos add.sig   add.mode     add.mu1     add.mu2     add.mu3     add.mu4
1   0 6728      2  0.3253068 -0.05632611 -0.13656087 -0.34259138 -0.45600212
2   0 6883      3  9.0963385  1.39688597 -2.43336304  1.10625151  0.05672361
3   0 6899      4  5.3871507  1.03125835  0.91000256 -1.73414760  0.69877344
4   0 7897      4 23.2958857  1.27028639 -0.32635179  0.95085732 -4.54660567
5   0 1271      0  0.0000000  0.04612361  0.03978521 -0.08060622  0.08770705
dom.sig  dom.mode    dom.mu1    dom.mu2    dom.mu3    dom.mu4
1       4  9.752713 1.33940947  0.9981518  2.0343131  1.6804636
2       4 33.455043 1.40182206 -2.0304856  5.0462396  1.3793230
3       2  0.114697 0.16038721  0.2982834 -0.1270506 -0.2547293
4       0  0.000000 0.09405818  0.2397407 -0.3175207 -0.2058939
5       4 13.728352 1.23633519 -1.1626640  2.7197024 -1.8577562
```



6.5. Parallel computing

The SNPs data is usually so huge that it may spend plenty of time processing the program. To save the computing time, it is desirable to adopt parallel computing. In the function ***bls.simple, bls.plink, bls.snpmat, gls.simple, gls.plink*** and ***gls.snpmat,*** the parameter *nParallel.cpu* controls the number of the CPU used to compute.

The following example shows how to set the parallel computing.

```
ret<-gls.simple(phe.out,snp.out,Y.prefix="Y",Z.prefix="Z",covar.names=c("X_1"),
options=list(nParallel.cpu=7) );
```