The **HiGWAS** Package

Version 0.5

1. Introduction

The HiGWAS 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 HiGWAS package. The first Bayesian Lasso named ***BLS*** model thereinafter 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 HiGWAS package. The outline of this guide is as follows:

Section 2: Installation

Section 3: Data Format

Section 4: Bayesian Lasso Model (***BLS***)

Section 5: Bayesian Group Lasso Model (***GLS***)

Section 6: Internal Details

Section 7: Reference

We refer to Li et al. 2011 [1] and Li et al 2015[2] 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 HiGWAS 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 HiGWAS, download the package file and type the appropriate command in Linux terminal window or R console window as follows.

1) Install *HiGWAS* in Linux terminal (command line)

```
$ git clone https://github.com/wzhy2000/HiGWAS.git
$ cd HiGWAS
$ R CMD INSTALL package
```

2) Install *HiGWAS* in R console window

There are two methods to install this package. The first method use *install.packages* function to install the package after the source downloading from the GitHub.

```
> install.packages("HiGWAS_0.50.tar.gz", repos = NULL, type="source");
```

The second method is to use *devtools* package to install the package in the GitHub directly. Comparing with *install.packages*, this method doesn't need to download the source code beforehand.

```
> library("devtools");
> install_github("wzhy2000/HiGWAS/package")
```

In the current version, we employ GPU to improve the computation speed of *GLS*. If CUDA library is installed in the system, but it can't be detected automatically and compiled with this package, please specify the CUDA library manually as follows:

```
$ cd HiGWAS
$ R CMD INSTALL --configure-args="--with-cuda-home=/usr/local/cuda-8.0" package
```

Or

```
> install.packages("HiGWAS_0.50.tar.gz", repos = NULL, type="source", configure.args = c(HiGWAS="--with-cuda-home /usr/local/cuda-8.0"))
```

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

```
> library(HiGWAS)
```

3. Data format

The HiGWAS 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,-0.946,40.0,42.0,43.0,48.0,50.0,…,18.8,14.2,11.5,15.9,22.2,…
2, 3, -0.8,41.0,40.0,38.0,39.0,NA,…,17.8,12.2,13.5,15.9,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. The package ***does not provide*** any quality control functions or complex imputations. For the imputation function, the package only can impute missing SNPs based on the genotype frequency.

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 SNP 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 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 30.6
137 2349 0 0 2 34.4
…
```

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. Bayesian Lasso Model (*BLS*)

This package implements two statistical models: the *BLS* model and *GLS* model. The difference between two models is whether the longitudinal phenotype can be processed. 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. This section will illustrate how to process the non-longitudinal phenotype with *BLS* model.

4.1 Statistical model [1]

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).

$$y_i = \mu + X_i^T\alpha + Z_i^T\beta + \xi_i^T a + \zeta_i^T d + \epsilon_i, \qquad i = 1, \dots, n, \tag{1}$$

where μ is the overall mean, X_i is the vector of binary discrete covariates, α 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, $a = (a_1, \dots, a_p)^T$ and $d = (d_1, \dots, d_p)^T$ 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. Preparing data

The package supports 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. We give two examples to do simulation functions as follows.

```
library(HiGWAS);

# Example 1: use the default parameters to make simulation data
bls.simulate("bls.phe.csv", "bls.gen.csv");

# Example 2: suppose 5 SNPs have additive or dominant effects.
sigsnp <- c(11, 22, 33, 444, 55);
# Simulate one test data with 1000 individual, 2000 SNPs
r.sim <- bls.simulate("bls.phe.csv", "bls.gen.csv",
     simu_grp=1,
     simu_n= 1000,
     simu_p= 2000,
     simu_snp_rho = 0.1,
     simu_rho    = 0.4,
     simu_sigma2  = 9,
     simu_mu     = 24,
     # 2 covariates, specify the coefficients
     simu_cov_effect = c( 0, 2 ),
     # 3 additive SNPs
     simu_add_pos   = c( sigsnp[1], sigsnp[2], sigsnp[3]),
     # The coefficients for each additive SNP
     simu_add_effect= c( 2.2, -2.5, 2.0 ),
     # 3 dominant SNPs
     simu_dom_pos   = c( sigsnp[3], sigsnp[4], sigsnp[5]),
     # The coefficients for each dominant SNP
     simu_dom_effect= c( 2.8, 2.0, -2.5 ),
     # The covariate range
     simu_cov_range=c( 0, 1),
     # The time range
     simu_t_range = c(-1, 1),
     debug=F );
```

After the calling of the simulation function, the phenotype file and genotype file are generated based on the simulation parameters.

4.3 Starting computation

The *BLS* model has three similar functions to analyze PLINK data, simple SNP data and SNP matrix data as follows.

```
> r.bls <- bls.simple(file.phe.long, file.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"));
```

The function ***bls.simple*** is designed to load files encoded by the Simple format, ***bls.plink*** is

intended to load PLINK files, but ***bls.snpmat*** is 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 3 functions. Four types of control parameters allow the users to supply the following information.

1) Response variables (*Y.name*) and covariate names (*covar.names*)
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 uses the simulation data to show how to do a computation using variable selection procedure only.

```
# Loading the phenotypic and genotypic data
tb.phe<-read.csv("bls.phe.csv", row.names=1);
tb.snp<-read.csv("bls.gen.csv");

# Calling variable selection only (refit=F)
r.bls  <-  bls.snpmat(tb.phe,  tb.snp,  Y.name="Y",  covar.names=c("X_1","X_2"),
fgwas.filter = F, refit = F );
```

After running above codes, the package outputs the parameters and optional items for the current data processing as follows:

```
[BLS SNPMAT] Procedure.
Checking the parameters ......
* Phenotypic Matrix:  1000 3
* SNP Matrix:   2000 1002
* Individuals:  1000
* SNPs:  2000
* Response Variable:  Y
* Covariate Columns:  X_1 X_2
* fGWAS Filter Used:  No
* Additive Effects Used:  Yes
* Dominant Effects Used:  Yes
* Refit Procedure:  No
* GPU Used:  No
Checking the optional items......
* Parallel Computing:  No, 0 CPU(s)
* p-value of fGWAS filter:  0.05
* Iteration of  Markov chain:  2000
* fBurnInRound:  0.3
* fRhoTuning:  0.095
* Debug Output:  No
Phenotypic data frame is converted to the matrix class.
Genetic Effect Analysis by BLS method......
```

Although the BLS model does MCMC sampling 2000 times, however the individuals and SNPs are not too large, this function calling only performs 7 minutes on the Intel Xeon E5 2620 CPU. If the SNP count or individual numbers are huge, the computation task will perform a long time.

4.4. Summarizing results

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

1) Significant SNPs estimated by the *fGWAS* method (See the details in the section 6)

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

The structure of the result object can be checked by the *str* function. For example, the result calculated in 4.3 contains the following items:

```
> str(r.bls)
Dotted pair list of 5
 $ varsel      : num [1:2000, 1:11] 1 1 1 1 1 1 1 1 1 1 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2000] "G0-SNP1" "G0-SNP2" "G0-SNP3" "G0-SNP4" ...
  .. ..$ : chr [1:11] "grp" "pos" "add.sig" "add.mu" ...
 $ varsel_cov  : num [1:3, 1:4] 1 0 1 19.182 0.0265 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "intercept" "X_1" "X_2"
  .. ..$ : chr [1:4] "cov.sig" "cov.mu" "cov.min" "cov.max"
 $ varsel_Qbest: num [1:2000, 1:8] 0.479 0.459 0.482 0.374 0.49 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2000] "G0-SNP1" "G0-SNP2" "G0-SNP3" "G0-SNP4" ...
  .. ..$ : NULL
 $ options     :List of 10
  ..$ nParallel.cpu   : num 0
  ..$ nPiecewise.ratio: num 0
  ..$ nMcmcIter       : num 2000
  ..$ fBurnInRound    : num 0.3
  ..$ fRhoTuning      : num 0.095
  ..$ fQval.add       : num 0.05
  ..$ fQval.dom       : num 0.09
  ..$ fgwas.cutoff    : num 0.05
  ..$ debug           : logi FALSE
  ..$ params          :List of 7
  .. ..$ Y.name      : chr "Y"
  .. ..$ covar.names : chr [1:2] "X_1" "X_2"
  .. ..$ refit       : logi FALSE
  .. ..$ gpu.used    : logi FALSE
  .. ..$ add.used    : logi TRUE
  .. ..$ dom.used    : logi TRUE
  .. ..$ fgwas.filter: logi FALSE
 $ elapsed     :Class 'proc_time'  Named num [1:5] 370.3 6.9 377.3 0 0
 - attr(*, "class")= chr "BLS.ret"
```

We can use the ***summary*** command or direct ***print*** command to show a brief result:

```
> summary( r.bls );

#OR

> print(r.bls);

# OR

> r.bls;
```

These 3 commands give same summary information illustrated in the following block.

```
> summary( r.bls );
--- Covariate Estimate in Varsel Procedure:
          Sig. Median CI.025 CI.975
intercept Yes 19.182 18.331 19.983
X_1       ---  0.026 -0.125  0.172
X_2       Yes  0.331  0.201  0.463
--- Variable Selection Result: 72 SNPs
Top 25 SNPs:
          Chr Pos Add.Sig Add.Median Dom.Sig Dom.Median    H2
G0-SNP4     1   4     ---      0.008     Yes      0.051 0.007
G0-SNP11    1  11     Yes      0.865     ---      0.006 3.713
G0-SNP22    1  22     Yes     -0.789     ---      0.004 3.090
G0-SNP24    1  24     Yes     -0.093     ---      0.004 0.043
G0-SNP33    1  33     Yes      0.682     Yes      0.680 3.591
G0-SNP35    1  35     ---     -0.005     Yes      0.165 0.068
G0-SNP55    1  55     ---     -0.004     Yes     -0.620 0.955
G0-SNP68    1  68     ---     -0.012     Yes     -0.063 0.010
G0-SNP79    1  79     Yes     -0.346     ---      0.007 0.594
```

4.5. Plotting figures

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

1) The Manhattan figure exported by the fGWAS method (*.fgwas.pdf) if *fGWAS* filter is enabled.

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 is required to call the function *plot*.

```
> plot (r.bls, fig.prefix="bls-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 optional parameters will be selected to variable selection. The figures of genetic effects are different depending on which model is 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. The following figure shows a result of variable selection in above example.

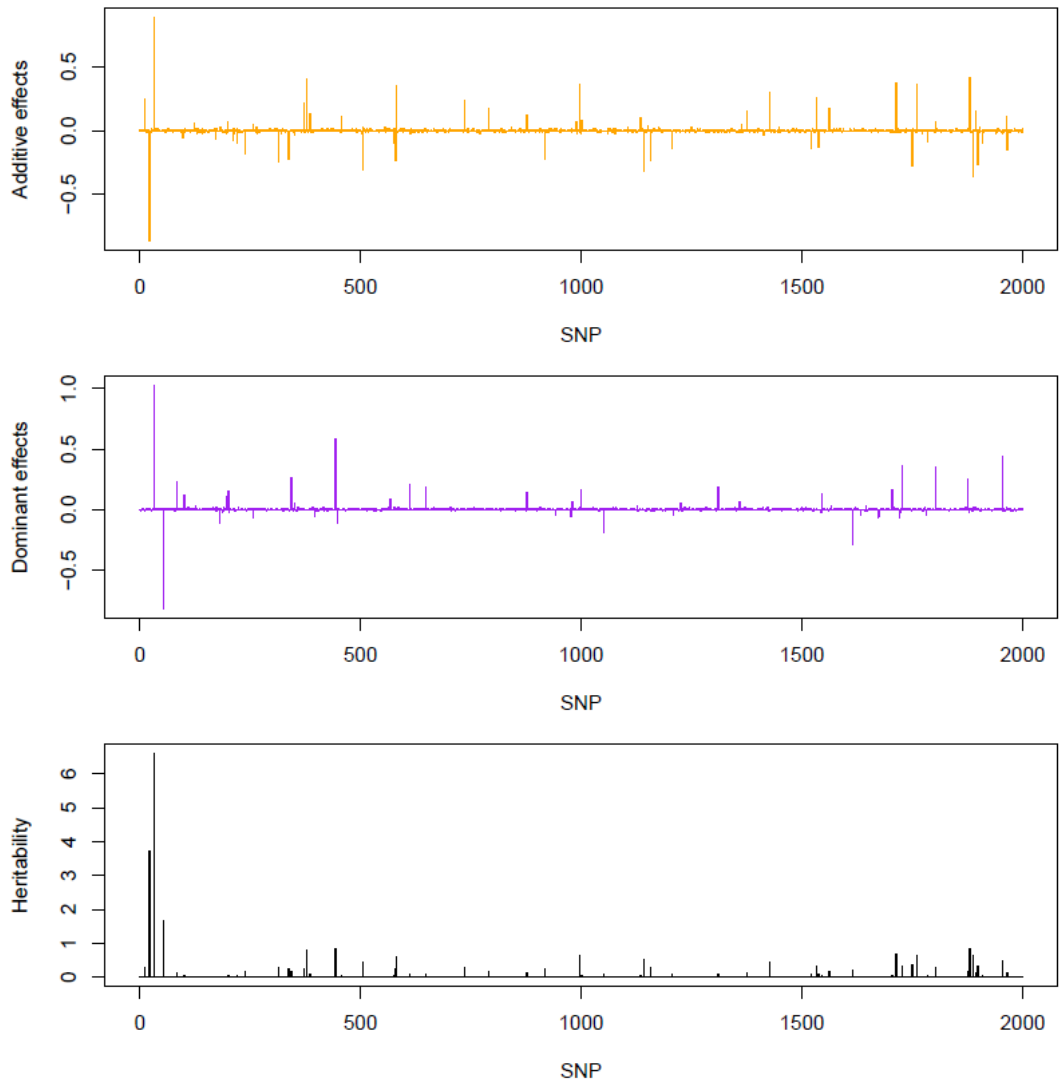


Figure 1: The simulation results of Bayesian Lasso. The plot includes additive effects, dominant effects and heritability for each SNP.

5. Bayesian Group Lasso Model (GLS)

The *GLS* model identifies joint genetic effects from the longitudinal phenotypic data that discover the dynamic patterns of the traits. This section introduces the statistical model and functions implemented in *GLS*.

5.1 Statistical model [2]

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

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

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

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.

5.2. Preparing data

Two types of genotypic data, PLINK format and simple CSV file described in 3.2.1, can be loaded by the GLS model, However, the *imputation* and *quality control* are supposedly finished according to the experiment requirement, this same requirements are also applicable for the *BLS* model.

Although PLINK has simple phenotypic data, the package use the user-defined CSV files to store the longitudinal phenotypic traits and covariates. We elaborated this format in 3.1.

The simulation function is provided for users who would like to try this package without any real data. The details of the simulation function are explained in the section 6.1. Here we show an example as follows.

```
library(HiGWAS)
#use the default parameters to make simulation data
gls.phe.out <- "gls.test.phe"
gls.snp.out <- "gls.test.snp"
cov_effect <- array(0, dim=c(2,4));
cov_effect[1,] <- c( 2.49, -1.135, 0.82, 0.425);
cov_effect[2,] <- c( -1.045, 2.320, 0.905,  0.535);
r.sim <- gls.simulate( gls.phe.out, gls.snp.out, simu_n=1000, simu_p=500,
   simu_cov_effect = cov_effect,
   simu_add_pos = c(50, 150, 250),
   simu_dom_pos = c(250, 350, 450),
   plink.format = TRUE );
```

5.3 Starting computation

Each statistical model has three simulation functions to analyze PLINK data, simple SNP data and SNP matrix data.

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

Although parameters for data source are different, the optional parameters are same for the overall 3 functions. Five types of control parameters allow the users to supply the following information.

1) The prefix of response variables (*Y.prefix*), measure time variable (*Z.prefix*), and covariate names (*covar.names*)
2) If the filter implemented by fGWAS is used to pre-select significant SNPs? (same as *BLS*)
3) If either additive or dominant effect is estimated? (same as *BLS*)
4) If only variable selection procedure is applied to data analysis? (same as *BLS*)
5) If GPU is available for the computation?

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 perform this computation including variable selection and refit procedure on CPU. If you do have GPU and installed with CUDA options, you can try it on GPU specifying "gpu.used=T".

```
if(r.sim $err==0)
{
  file.plink.bed <- r.sim$file.plink.bed
  file.plink.bim <- r.sim$file.plink.bim
  file.plink.fam <- r.sim$file.plink.fam

 r.gls <- gls.plink(gls.phe.out, file.plink.bed, file.plink.bim, file.plink.fam,
   Y.prefix="Y", Z.prefix="Z", covar.names=c("X_1", "X_2"),
   fgwas.filter = F,
   gpu.used=F,
   refit=T );
}
```

When the *GLS* model is running, the above codes will output the following information, including the calling parameters, optional parameters and running log.

```
* Phenotypic Data File:  gls.test.phe
* PLINK BED File:  gls.test.snp.bed
* PLINK BIM File:  gls.test.snp.bim
* PLINK FAM File:  gls.test.snp.fam
* PLINK Command:
* Force Split by PLINK Command:  FALSE
* Response Variable:  Y
* Time Variable:  Z
* Covariate Columns:  X_1 X_2
* fGWAS Filter Used:  No
* Additive Effects Used:  Yes
* Dominant Effects Used:  Yes
* Refit Procedure:  Yes
```

```
* GPU used:  No
Checking the optional items......
* Parallel Computing:  No, 0 CPU(s)
* Piecewise Ratio:  0
* p-value of fGWAS filter:  0.05
* Iteration of  Markov chain:  2000
* fBurnInRound:  0.3
* fRhoTuning:  0.095
* Debug Output:  No
Genetic Effect Analysis by BLASSO/GLASSO method......
* Final LASSO calling.
Round=0 sigma2=22.2 rho=0.4 tmp5=0.32 lambda2=17916,5.6,17730,0.3 mu=16.7, -1.7, 4.7
Round=1 sigma2=21.0 rho=0.4 tmp5=0.39 lambda2=269.2,0.26,285.9,2.1 mu=16.1, -1.5, 4.4
……
```

5.4. Summarizing results

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 its effects estimated in the variable selection procedure

4) Covariates coefficients estimated in the refit procedure

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

```
summary( r.gls );

# OR

print(r.gls)

# OR

r.gls
```

Below is an example calculated by the section 5.3, which shows the significant SNPs in the variable selection and refit procedure.

```
> r.gls
--- Covariate estimated in variable selection procedure:
           cov.sig         L2
Intercept        1      4.373
X_1              3      3.156
X_2              3      2.618
--- Variable selection result: 84 SNPs
Top 25 SNPs:
          chr pos add.sig    add.L2 dom.sig    dom.L2
G0-SNP31    1  31       2 0.3340733       0 0.0000000
G0-SNP36    1  36       1 0.1490189       1 0.1677417
G0-SNP46    1  46       1 0.1851682       0 0.0000000
G0-SNP50    1  50       4 2.1160388       0 0.0000000
……
--- Covariate estimated in refit procedure:
          cov.sig         L2
Intercept       4     13.519
X_1             3      3.097
X_2             3      2.575
--- Refit result: 5 SNPs
          chr pos add.sig    add.L2 dom.sig    dom.L2
```

```
G0-SNP50    1  50       4 2.7678745       2 0.5774434
G0-SNP150   1 150       4 4.6107338       1 0.1951339
G0-SNP250   1 250       3 2.2203017       4 3.1964101
G0-SNP450   1 450       1 0.1499482       4 5.7511103
G0-SNP350   1 350       0 0.0000000       4 4.1117220
```

The *L2* in the output indicate the L2 norm of Legendre Polynomials, namely,

$$\boldsymbol{L2 = \sqrt{r_0^2 + r_1^2 + r_2^2 + r_3^2}}$$

Where $\boldsymbol{r_0, r_1, r_2, r_3}$ are the coefficients of 4th-order of Legendre Polynomials. *cov.sig*, *add.sig* and *dom.sig* indicate how many coefficients are not zero in the Polynomials, namely, these items have genetic effects. The actual coefficients can be accessed via the list item *r.gls$varsel_add*, *r.gls$varsel_dom*, *r.gls$refit_add*, and *r.gls$refit_dom*, for example:

```
> head(r.gls$refit_add)
          grp pos add.r0 add.r1 add.r2 add.r3 add.mu.L2    add.mu0      add.mu1     add.mu2
G0-SNP50    1  50      1      1      1      1 2.7678745  1.0524758  0.641499821  0.74027791
G0-SNP150   1 150      1      1      1      1 4.6107338 -0.8393580 -1.174897457  4.26945967
G0-SNP250   1 250      1      1      1      0 2.2203017 -1.8972614 -0.309559186  1.11099585
G0-SNP350   1 350      0      0      0      0 0.0000000  0.1133273  0.008062415  0.21199497
G0-SNP450   1 450      1      0      0      0 0.1499482  0.1499482  0.119225400 -0.07895931
              add.mu3 add.min.L2     add.min0    add.min1    add.min2   add.min3 add.max.L2
G0-SNP50  -2.36514073 2.92547192  0.923590248  0.42475979  0.47278033 -2.7021149  2.7124500
G0-SNP150 -0.97245792 4.52553165 -0.961948698 -1.38765165  4.00258185 -1.2684056  4.7304871
G0-SNP250 -0.07239225 2.25350504 -2.024221072 -0.51781810  0.84420282 -0.3987613  2.2484573
G0-SNP350  0.03159078 0.00000000 -0.009866938 -0.20170599 -0.05722082 -0.2810902  0.0000000
G0-SNP450 -0.29458177 0.02820712  0.028207119 -0.08801847 -0.35823128 -0.6305424  0.2799885
            add.max0   add.max1  add.max2    add.max3
G0-SNP50   1.1790590  0.8602674 1.0027722 -2.05465144
G0-SNP150 -0.7111890 -0.9578569 4.5305398 -0.65455121
G0-SNP250 -1.7680954 -0.1031922 1.3851896  0.23778264
G0-SNP350  0.2369578  0.2242946 0.4760993  0.33650067
G0-SNP450  0.2799885  0.3364773 0.1879302  0.02307492
```

add.r0 indicate r_0 has genetic effect(not zero), and then *add.mu0* , *add.min0* and *add.max0* are the mean value, minimal value and maximum value of MCMC sampling, *add.r1*, *add.r2* and *add.r3* represents the r_1, r_2 and r_3 and so on.

5.5. Plotting figures

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

1) The Manhattan figure exported by the fGWAS method if *fgwas.filer* is used (*.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 *GLS* result object is required to call the function *plot.*

```
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 fGWAS filter is not involved in this function calling, so plot

function cannot make any Manhattan figure.

The figures of genetic effects are different depending on which model is used. The *GLS* model outputs L2 norm of four coefficients in Legendre Polynomials as genetic effects for each significant SNP. The following figures illustrate the results of variable selection. The higher loci have significant additive or dominant effects.

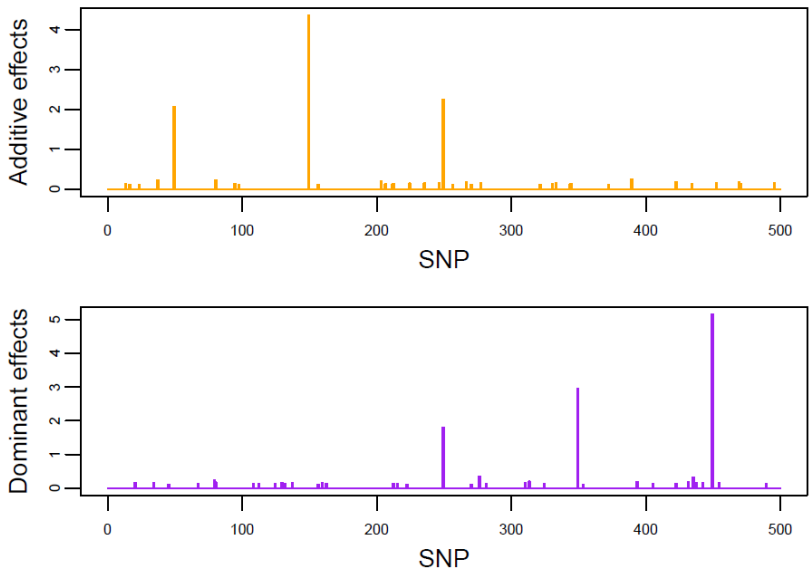
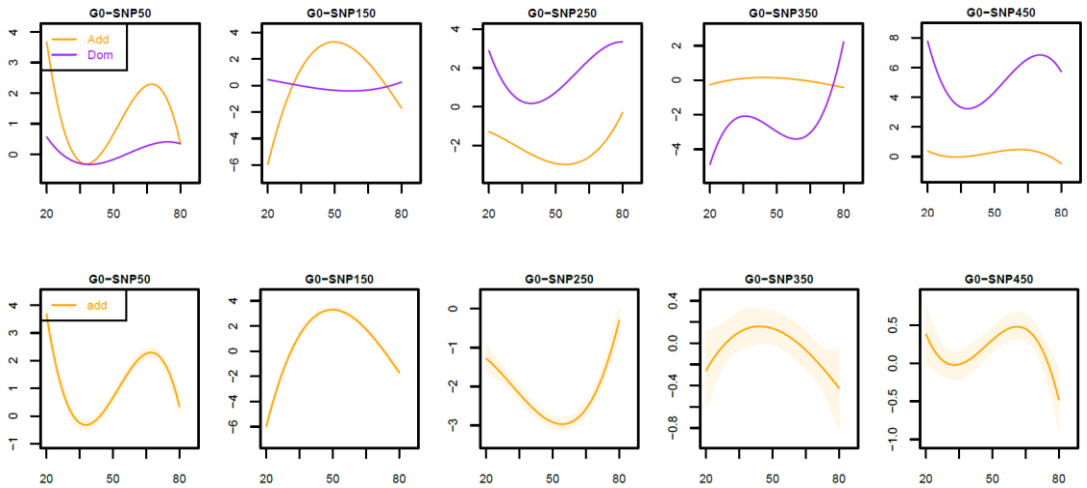


Figure 2: The simulation results of Bayesian Group Lasso. The plot illustrates L2-norm of additive effect and dominant effect for each SNP.

For the refit procedure, *GLS* model can output curves indicating time varying additive effects and dominant effects for each significant SNP as follows:



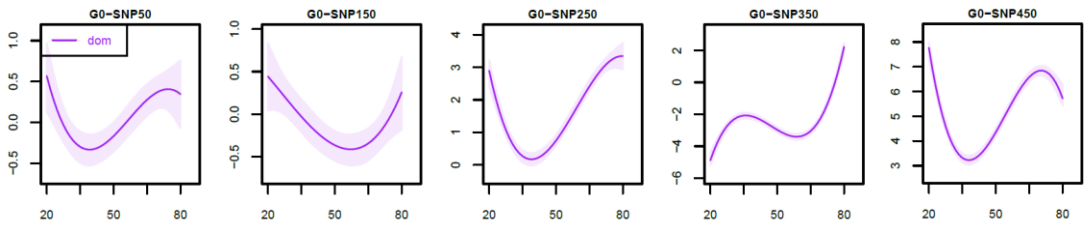


Figure 3: The time-varying effects detected by Bayesian Group Lasso. The first row shows the additive and dominant effects of 5 SNPs. The middle row only shows additive effects with 95% CI range. The last row only shows dominant effects with 95% CI range.

Please notice the measure times in this figure are normalized to the range between -1 and 1. The normalization is according to the formula:

$$t = \frac{\mathrm{Z} - \max(\mathrm{Z})}{\mathrm{Z} - \min(\mathrm{Z})} * 2 - 1$$

Where Z is the original measure times, t is normalized value and applied to Legendre Polynomials.

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 which has the same structure as real data. All pre-defined parameters can be customized. The following two tables and two boxes show the simulation parameters of the *BLS* model.

```
bls.simulate( file.phe.out, file.snp.out,
    simu_grp = 1, simu_n = 500, simu_p = 1000,
    simu_snp_rho = 0.1, simu_snp_missing = 0.002,
    simu_rho = 0.4, simu_sigma2 = 3, simu_mu = 26,
    simu_cov_range = c(0,1),
    simu_cov_effect = c(0, 2),
    simu_add_pos = c(100, 200, 300),
    simu_add_effect = c(2.2, -2.5, 2),
    simu_dom_pos = c(300, 500, 700),
    simu_dom_effect = c(2.8, 2, -2.5),
    simu_t_range = c(-1, 1),
    plink.format = FALSE, debug = FALSE)
```

Table1： Simulation Parameters of *BLS* 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	Float, the overall mean of phenotypic data
simu_add_pos	Vector, positions of the significant SNPs with additive effects
simu_add_effect	Vector, additive effects of significant SNPs.
simu_dom_pos	Vector, positions of the significant SNPs with dominant effects
simu_dom_effect	Vector, dominant effects of significant SNPs.
simu_cov_range	Vector, the range of covariates
simu_covar_effect	Vector, the effects of covariates
simu_t_range	Vector, the range of measure time points

```
gls.simulate (file.phe.out, file.snp.out,
    simu_grp = 1, simu_n = 500, simu_p = 1000,
    simu_snp_rho = 0.1, simu_snp_missing = 0.002,
    simu_rho = 0.4, simu_sigma2 = 16,
    simu_mu = c(13.395, -3.08, 1.875, -3.195),
    simu_cov_range = c(-1, 1),
    simu_cov_effect = array(c(0, 0, 0, 0), dim = c(1, 4)),
    simu_add_pos = c(1, 2, 3),
    simu_add_effect = array(c(1.04, 0.885, -2.055, 0.545, 1.17, -0.2, 0.74, -4.715, 1.4, -2.25, 1, 0), dim = c(3, 4)),
    simu_dom_pos = c(3, 4, 5),
    simu_dom_effect = array(c(1.49, -2.135, 4.82, 1.425, 1.045, 1.32, 1.905, 1.535, 1.265, -1.225, 2.71, -1.96), dim = c(3, 4)),
    simu_z_range = c(20, 80),
    simu_z_count = c(5, 12),
    plink.format = FALSE, debug = FALSE)
```

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_pos	Vector, positions of the significant SNPs with additive effects
simu_add_effect	Matrix, additive effects of significant SNPs where the first column represents positions of these SNPs
simu_dom_pos	Vector, positions of the significant SNPs with dominant effects
simu_dom_effect	Matrix, dominant effects of significant SNPs where the first column represents

	positions of these SNPs
simu_cov_effect	Matrix, effects of covariates
simu_cov_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. The following example shows how to generate data for the *GLS* model.

```
a_effect <- array(0, dim=c(3,4));
d_effect <- array(0, dim=c(3,4));
cov_effect <- array(0, dim=c(2,4));
sigsnp <- sample(1:100)[1:5];
a_effect[1,]<-c( 1.04, 0.885, -2.055, 0.545);
a_effect[2,]<-c( 1.17, -0.20, 0.74, -4.715);
a_effect[3,]<-c( 1.40, -2.25, 1.00,  0.00);
d_effect[1,]<-c( 1.49, -2.135, 4.82, 1.425);
d_effect[2,]<-c( 1.045, 1.320, 1.905,  1.535);
d_effect[3,]<-c( 1.265, -1.225, 2.710, -1.96);
cov_effect[1,]<-c( 2.49, -1.135, 0.82, 0.425);
cov_effect[2,]<-c( -1.045, 2.320, 0.905,  0.535);
simu.data <- gls.simulate("gls.test.phe", "gls.test.snp",
      simu_n= 5000, simu_grp=1, simu_p=2000,
      simu_snp_rho=0.4, simu_rho=0.1, simu_sigma2= 4,
      simu_mu= c(13.395, -3.08, 1.875, -3.195),
      simu_cov_effect = cov_effect,
      simu_cov_range  = c(-1,1),
      simu_add_pos    = c( sigsnp[1], sigsnp[2], sigsnp[3] ),
      simu_add_effect = a_effect,
      simu_dom_pos    = c( sigsnp[3], sigsnp[4], sigsnp[5] ),
      simu_dom_effect = d_effect,
      simu_z_range    = c(30,60),
      simu_z_count = c(5,12),
      plink.format=T, debug=F);
```

Please notice in above example, we use "*plink.format*=T" to generate the genotype file in the PLINK format.

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 non-significant SNPs could avoid the problem. This package provide a SNP filter using the reduced fGWAS model[3] which compares the likelihood between the regression only with covariates and the regression with the SNP effects. In the variable selection step, this filter can be enabled by specifying the parameter *fgwas.filter*. For examples:

```
r.bls <- bls.snpmat(tb.phe, tb.snp,
   Y.name="Y", covar.names=c("X_1","X_2"),
   fgwas.filter = T, refit = T,
```

```
   options=list(nParallel.cpu=10, fgwas.cutoff=0.01) );

r.gls <- gls.plink(gls.phe.out, file.plink.bed, file.plink.bim, file.plink.fam,
   Y.prefix="Y", Z.prefix="Z", covar.names=c("X_1", "X_2"),
   fgwas.filter = T, gpu.used = T, refit = F,
   options=list(nParallel.cpu=10, fgwas.cutoff=0.01) );
```

The filter results are stored in the result object, which can be viewed by two methods: summarizing the result object or plotting the Manhattan figure.

1) The *summary* command shows the significant SNPs estimated by fGWAS method, for example:

```
> r.gls
> r.gls
--- Significant SNPs estimated by fGWAS method: 22 SNPs
          SNP.IDX CHR POS N.miss       MAF    L.Ratio           pv
G0-SNP450     450   1 450      0 0.4948757 636.109268 7.423823e-139
G0-SNP250     250   1 250      0 0.4875722 258.853423  6.175802e-57
G0-SNP350     350   1 350      0 0.4760278 114.087385  1.683601e-25
G0-SNP50       50   1  50      0 0.4900459  36.010975  1.514663e-08
G0-SNP324     324   1 324      0 0.4972906  17.792767  1.368831e-04
……
```

2) A Manhattan figure generated by the *plot* command gives –log10 p-values for each SNP, for example:

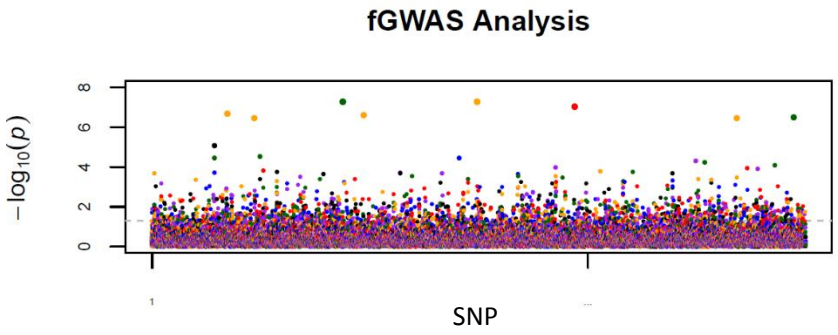


Figure 4: The Manhattan plot of simulation.

3) The full results can be accessed by the list item *fgwas*, for example*:*

```
> head(r.gls$fgwas)
        SNP.IDX CHR POS N.miss       MAF   L.Ratio         pv
G0-SNP1       1   1   1      0 0.4977110 3.2246026 0.19942814
G0-SNP2       2   1   2      0 0.4949525 4.6190078 0.09931051
G0-SNP3       3   1   3      0 0.4998239 1.2693426 0.53010971
G0-SNP4       4   1   4      0 0.4812185 0.9063776 0.63559811
G0-SNP5       5   1   5      0 0.4901984 3.4078455 0.18196830
G0-SNP6       6   1   6      0 0.4940721 6.8285321 0.03290055
```

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 are estimated by the MCMC

algorithm.

The results of variable selection have these features:

1) Variable Selection Result stored in the list item *varsel_add*, *varsel_dom* in *GLS* and *varsel* in *BLS*.

2) The *summary* command shows the additive effects and dominant effects of the top 25 SNPs as the illustrated in the section 5.4 and 4.4.

3) The *plot* command draws a figure containing addictive effects, dominant effects, and heritability of selected SNPs.

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 from variable selection are included in the final model.

The results of refit object have these features:

1) Variable Selection Result stored in the list item *refit_add*, *refit_dom* in *GLS* and *refit* in *BLS*.

2) The *summary* command shows the additive effects and dominant effects of the significant SNPs as the illustrated in the section 5.5 and 4.5.

3) The *plot* command draws a figure containing addictive effects, dominant effects, and heritability in *BLS* or containing genetic curves in *GLS* for the significant SNPs.

6.5. GPU computing

GPU has powerful parallel capability to improve the computation speed, especially for MCMC iteration in *GLS*. In this package, we have implemented some *for-loop* computation on GPU for the *GLS* model. If users have successfully installed this package with the CUDA library, R console outputs the GPU information after the package is loaded, for example:

```
> library(HiGWAS)
2 GPU card(s) is/are available for the GLS model.
```

After this message, user need to specify the parameter *gpu.used* to start the computation on GPU.

```
r.gls <- gls.plink(gls.phe.out, file.plink.bed, file.plink.bim, file.plink.fam,
   Y.prefix="Y", Z.prefix="Z", covar.names=c("X_1", "X_2"),
   fgwas.filter = F,
   gpu.used = T,
   refit = T );
```

6.6. Performance comparison of CPU computing and GPU computing

GLS is still a computing intensive process, even the C/C++ codes are employed to improve speed on GPU. We used the simulation data to compare multiple experiments on CPU (Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz) and GPU (NVIDIA TITAN X (Pascal), 12GB memory), and obtained the following performance results in Figure 5. Panel E indicates

GPU is faster than CPU at least 5-fold at the same sample size and the performance become better rapidly with the increase of sample size. However, Panel F indicates SNP size reduces the performance a little bite. Here we use the regression to evaluate the computing time for CPU and GPU as the follows:

CU time = exp(-14.5794 + 1.5311*log(Sample) + 0.8878*log(SNP))

GPU time = exp(-11.5997 + 0.7051*log(Sample) + 0.9711*log(SNP))

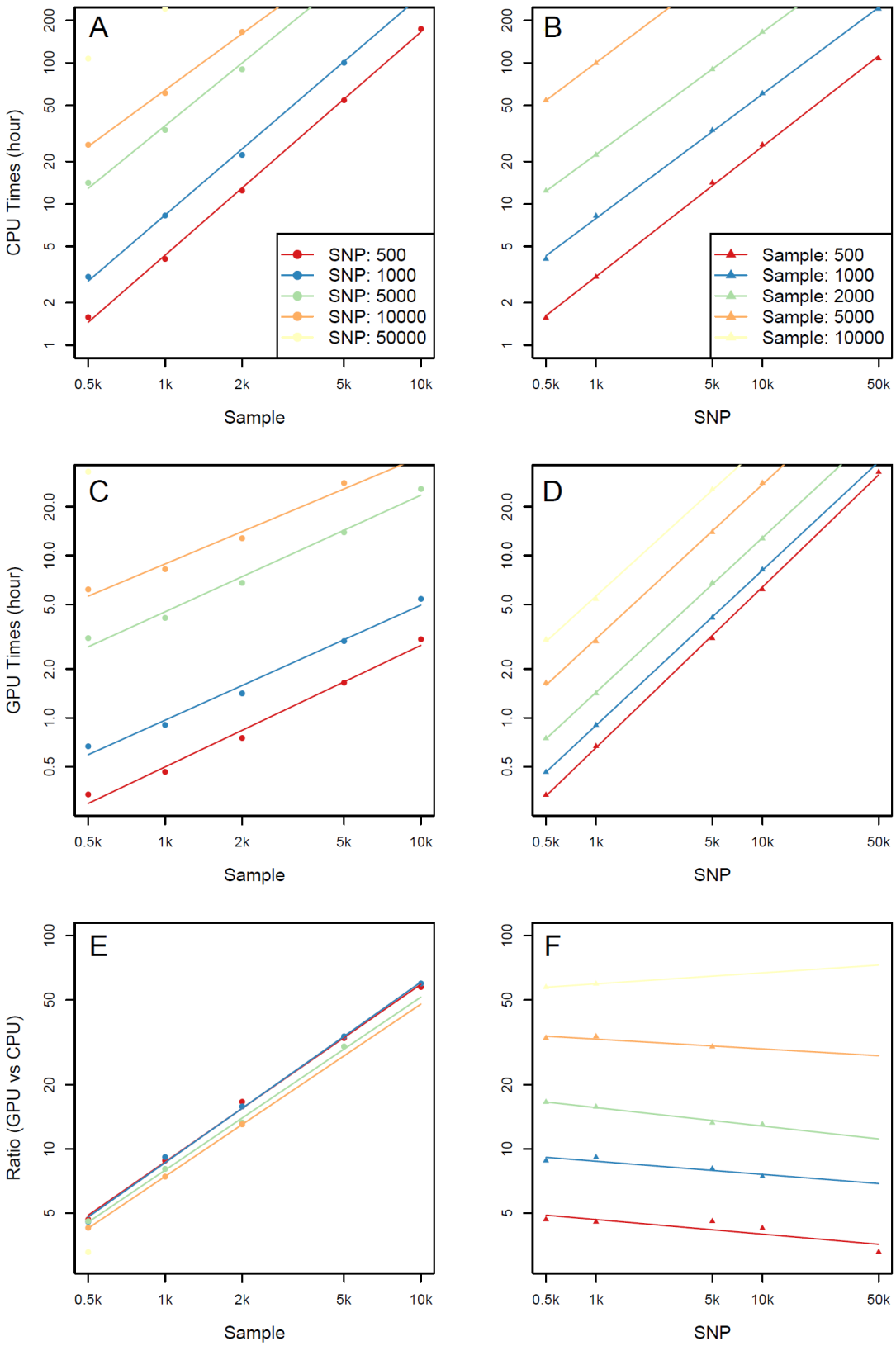


Figure 5: Comparison of CPU and GPU performance at the multiple combinations of sample size and SNP size. The X and Y-axis are log scale.

7 Reference

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).

3. Das, K., Li, J., Wang, Z., Tong, C., Fu, G., Li, Y., ... & Wu, R. (2011). A dynamic model for genome-wide association studies. Human genetics, 129(6), 629-639.