SODABayes

User's Guide

Version 1.10

Livestock Gentec Centre
Department of Agriculture, Food, and Nutritional Science
University of Alberta
Edmonton, Alberta, Canada

Liuhong Chen & Graham Plastow

December 20, 2021

Contents

1 Introduction

SODABayes is a program for implementing Bayesian linear regression models for whole-genome analysis of quantitative traits using a novel subsetted orthogonal data augmentation approach. The approach enables parallel implementation of Gibbs sampling algorithms for the estimation of marker effects. The program was written in Fortran 2018 with OpenMP API.

Current version supports a variety of Bayesian models including BayesA, BayesB, BayesC, BayesCπ, BayesR, and BayesRc.

2 Download SODABayes

SODABayes is free for academic use. The source code can be downloaded from:

https://github.com/liuhong-chen/SODABayes

The source code folder contains 7 files:

SODABayes.f90 – the main program

globals.f90 – a module that defines global variables

parallelrng.f90 – parallel random number and random variable generator

utilities.f90 – functions and subroutines that can be used by other modules

useroptions.f90 – a module that handles various user defined options

soda.f90 – a module that generates the orthogonal augmenting matrix

models.f90 – contains various models using either regular or SODA algorithms

The source code can be compiled using a Fortran compiler that supports Fortran 2018 and OpenMP 4.5 languages. For example,

To use intel Fortran compiler under Linux, type:

ifort -o sodaBayes -qopenmp globals.f90 parallelrng.f90 utilities.f90 useroptions.f90 soda.f90 models.f90 SODABayes.f90

or to use gfortran compiler, type:

gfortran -o sodaBayes -fopenmp globals.f90 parallelrng.f90 utilities.f90 useroptions.f90 soda.f90 models.f90 SODABayes.f90

Executables for windows x64 and Linux systems can be downloaded from the bin folder.

The example folder contains several examples.

3 Run SODABayes

3.1 Options

The program accepts various user supplied options dealing with job handling, input data files, Bayesian methods and hyper-parameters, data augmenting, MCMC sampling, prediction, and output files. The input format for a regular option is: -key [value], and for a logical option, 'value' is omitted.

3.2 Job handling

Options can be placed in the command line or a parameter file.

For example, the following command line runs BayesCpi using the default SODA approach:

```
./sodaBayes -bfile simdata1_train -method BayesCpi -scale_va 0.01 -scale_ve 50 -df_va 4.5
-df_ve 4.5 -numit 10000 -burnin 2000 -nthreads 10 -nblocks 10 -out sodaBayesCpi_simdata1
```

The same procedure can be run from a parameter file using the following command line:

```
./sodaBayes example.par
```

The example.par file contains:

```
BEGIN example
  -bfile simdata1_train
  -method BayesCpi
  -scale_va 0.01
  -scale_ve 50
  -df_va 4.5
  -df_ve 4.5
  -numit 10000
  -burnin 2000
  -nthreads 10
  -nblocks 10
  -out sodaBayesCpi_simdata1
END example
```

In this case, the suffix of the parameter file needs to be '.par'. When a different suffix is used, a key for the parameter file needs to be specified in the command line as: -parfile [parfile_name]. The parameter file contains sections enclosed by pairs of BEGIN and END statements. The parameter file can contain multiple job sections each enclosed by a pair of BEGIN and END statement. The program will run the jobs sequentially. The job name after the BEGIN and END statements is optional. A job name can also be specified using option: -job_name [job_name].

When the same option key appears in both command line and parameter file, the value for that option in the parameter file is ignored. This is useful when running different options using the same template parameter file. For example, to run BayesB method with the above example.par file, we can simply type:

./sodaBayes example.par -method BayesB -out sodaBayesB_simdata1

3.3 Input data files

The program uses plink binary genotype format, and the bed, bim, and fam files are mandatory. The prefix for these three files needs to be placed after the key -bfile. Alternative phenotype file can be used by using the -pheno option. When -pheno is used, either -pheno_name or -pheno_ncol is required to specify the name or the column number of the phenotype. When -pheno_name is used, a header line that contains the name of the phenotype column is required and must start with a $ symbol. A headline is not required if -pheno_ncol is used.

For example: To use y1 in the following pheno.txt as phenotype, choose -pheno pheno.txt -pheno_name y1; or -pheno pheno.txt -pheno_ncol 2.

pheno.txt

$id	y1	y2
1	64.22	0
2	88.88	1
3	64.02	0
4	68.51	0
5	61.08	0
6	70.34	0
7	68.57	0
8	76.43	0
9	73.67	1
10	71.49	0

3.4 Bayesian methods and hyper-parameters

The -method option is used to select a Bayesian method from BayesA, BayesB, BayesC, BayesCpi, BayesR, and BayesRc.

The scale and degree of freedom for the inverse scaled chi-square distribution for the additive SNP variance can be set via -scale_va and -df_va, and -scale_ve and -df_ve options are used to set the hyper-parameters for the residual variance.

For BayesB and BayesC, the fixed pi value for the proportion of zero effects can be set via -pi option (e.g., -pi 0.99).

For BayesR and BayesRc, the number of mixture distributions are set by -nmix, and the proportional variances are set by -mix (e.g., -nmix 4 -mix 0,0.0001,0.001,0.01). The hyper-parameter, alpha for the Dirichlet (BayesR and BayesRc) or the Beta distribution (BayesCpi) used to sample pi values can be set via the -alpha option. The default value for alpha is 1.

For BayesRc, a file that contains group information for each SNP is required. It can be specified use -grpfile [group file]. The file contains two columns. The first column is SNP ID, and the second column is the group ID.

3.5 Data augmenting

A few options are related to the data augmenting approach. The first two are -SODAOn and -SODAOff (no key value), which turns data augmenting on and off, respectively. The default setting is to run SODA. So, -SODAOn can be omitted if the purpose is to run a job with the SODA approach. To run a regular MCMC sampling use -SODAOff.

The next is to define the number of blocks which can be set by using -nblocks n. The number of SNPs in a block will be the total number of loci divided by the number of blocks. Some blocks may have an extra SNP is it is not divisible. Alternatively, one can use -block_size m to set the number of SNPs in a block. The number of blocks will be calculated as total number of loci divided by block_size. If it is not divisible, an additional block will be added for the remaining SNPs.

By default, the SNPs are assigned to blocks sequentially based on their order in the genotype file. To assign SNPs randomly, the option -rblocks (no key value) can be used.

The number of threads to run a parallel MCMC sampling can be set using -nthreads. If it is not set, the default is to use maximum threads available.

3.6 MCMC sampling

The total number of MCMC cycles and burn-in period can be set via -numit and -burnin, respectively. The thinning rate of the MCMC samples can be set via -thin. The default value for -thin is 1, i.e., without thinning. The seed value for random number generator can be set via -seed. If -seed is not used a number from system_clock() is used to set the seed.

3.7 Prediction

The default setting of the program is to run MCMC sampling. To run prediction using existing solution files, use -predict (no key value). Three files need to be supplied, with the model file

containing the general mean, the freq file containing the allele frequencies for all the markers, and the effects file containing additive snp effects. These files are set via -model [model file], -freq [freq file], and -effects [effect file].

3.8 Output files

Prefix for the output file names can be specified via -out option.

For MCMC, the output includes .log file, .hyp file, .mod file, .frq file, .eff file, and .gv file.

The .log file contains the options used to run the job, time the job started, time the job ended, and total time used to run the job.

The .hyp file contains intermediate outputs for several model parameters, including iteration number, model size, current estimates for general mean and residual variance, and other parameters depending on which Bayesian method was chosen.

The .mod file contains posterior solutions for general mean, residual variance, common marker additive variance (for models BayesC, BayesCpi, BayesR, and BayesRc), posterior mean number of SNPs included in the model, pi values for the mixture distributions (BayesCpi, BayesR, and BayesRc), and total variance explained by each mixture distribution.

The .frq file contains SNP ID and allele frequency for each SNP.

The .eff file contains SNP ID, snp additive effects, SNP specific variances (BayesA and BayesB), and posterior inclusion probabilities for the SNP belonging to each of the mixture distribution.

The .gv file contains animal ID, and genomic estimated breeding values for training animals.

For prediction, the output includes a .log file and a .gv file.

4 Examples

Example 1: run SODABayesR for MCMC sampling and prediction

```
BEGIN SODABayesR_Example_MCMC
  -bfile ../data/simdata1_train
  -method BayesR
  -scale_va 50
  -scale_ve 50
  -df_va 4.5
  -df_ve 4.5
  -numit 10000
  -burnin 2000
  -thin 1
  -nmix 4
  -mix 0.0,0.0001,0.001,0.01
  -nthreads 10
  -nblocks 10
  -seed 123456789
  -out bayesr_mcmc_example_1
END SODABayesR_Example_MCMC

BEGIN SODABayesR_Example_Prediction
   -bfile ../data/simdata1_pred
   -predict
   -model   bayesr_mcmc_example_1.mod
   -freq    bayesr_mcmc_example_1.frq
   -effects bayesr_mcmc_example_1.eff
   -out pred_bayesr_mcmc_example_1
END SODABayesR_Example_Prediction
```

Example 2: run regular BayesR MCMC sampling

```
BEGIN BayesR_Regular_MCMC
  -bfile ../data/simdata1_train
  -method BayesR
  -scale_va 50
  -scale_ve 50
  -df_va 4.5
  -df_ve 4.5
  -numit 10000
  -burnin 2000
  -thin 1
  -nmix 4
  -mix 0.0,0.0001,0.001,0.01
  -SODAOff
  -seed 123456789
  -out bayesr_mcmc_example_2
END BayesR_Regular_MCMC
```

Example 3: run prediction only

```
BEGIN BayesR_Example_Prediction
   -bfile ../data/simdata1_pred
   -predict
   -model   bayesr_mcmc_example_2.mod
   -freq    bayesr_mcmc_example_2.frq
   -effects bayesr_mcmc_example_2.eff
   -out pred_bayesr__example_3
END BayesR_Example_Prediction
```

Example 4: Run SODABayesRc MCMC sampling and prediction

```
BEGIN BayesRc_Example_MCMC
  -bfile ../data/simdata1_train
  -method BayesRc
  -grpfile ../data/group.txt
  -scale_va 50
  -scale_ve 50
  -df_va 4.5
  -df_ve 4.5
  -nmix 4
  -mix 0,0.0001,0.001,0.01
  -numit 10000
  -burnin 2000
  -thin 10
  -nthreads 10
  -nblocks 10
  -seed 123456789
  -out bayesrc_mcmc_example_4
END BayesRc_Example_MCMC

BEGIN BayesRc_Example_Prediction
   -bfile ../data/simdata1_pred
   -predict
   -model   bayesrc_mcmc_example_4.mod
   -freq    bayesrc_mcmc_example_4.frq
   -effects bayesrc_mcmc_example_4.eff
   -out pred_bayesrc_mcmc_example_4
END BayesRc_Example_Prediction
```