'LEP' Package for Joint Analysis of Individual-level and Summary-level GWAS Data by Leveraging Pleiotropy

Mingwei Dai [1,2], Xiang Wan [3], Hao Peng [4], Yao Wang [1], Yue Liu [2],
Jin Liu [6], Zongbe Xu [1], Can Yang [2]
[1] School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, China.
[2] Department of Mathematics, Hong Kong University of Science and Technology, Hong Kong.
[3] ShenZhen Research Institute of Big Data, ShenZhen, China.
[4] School of Business Administration, Southwestern University of Finance and Economics, Chengdu, China.
[5] Xiyuan Hospital of China Academy of Chinese Medical Sciences, Beijing, China.
[6] Centre for Quantitative Medicine, Duke-NUS Medical School, Singapore.

April 16, 2018

1 Overview

This vignette provides an introduction to the 'LEP' package. LEP is a statistical approach for joint analysis of individual-level and summary-level GWAS data by leveraging pleiotropy in Genome Wide Association Studies. This package provides computationally efficient and user friendly interface to fit and evaluate the LEP model. It accepts both the R-type data and binary plink files.

The package can be loaded with the command:

```
R> library("LEP")
```

This vignette is organized as follows. Section 2.1 discusses how to fit LEP in various settings. Section 2.2 show how to evaluate the performance in terms of cross validation. Section 2.3 shows how to predict by the well-trained model.

2 Workflow

In this vignette, three different simulated data sets are used for demonstration. (1). R-type D1 = {X0, y0, P0} are genotype, phenotype and p-values, they have no information (SNP names) for the SNPs; (2) R-type D2= {X, y, P} are the counterparts, but they contain the information for the SNPs; (3) the genotyp data in the plink format are 'sim0.bed', 'sim0.fam', 'sim0.bim', the p-values stored in {P} are with SNP information. For the simulation data, $\{X, X0\}$ are both $N \times M$ matrix, where $N = 1000$ is the sample size and $M = 3000$ is the number of SNPs; $\{y, y_0\}$ are both $N \times 1$ vector; $\{P, P0\}$ are both $M \times K$ matrix, where $M = 3000$ is for the number of SNPs, $K = 3$ is for the nubmer of GWAS.

The R-type data used in this package could be loaded by the command.

```
R> data(DB)
```

The binary plink files could be accessed by

```
R> plinkfile <- gsub(".bim","",system.file("extdata", "sim0.bim", package = "LEP"))
```

2.1 Fitting the LEP

R package `LEP` provides flexible statistical framework and automatically adjusts its model structure based on the provided data. The `LEP` model could be fitted in the following three ways.

2.1.1 R-type data with no SNPs' information

In this subsection, the matrices of genotype data and p-values, which have not any information for SNPs, are used. It requires that

```
R> nrow(X0) == length(y0)

[1] TRUE

R> ncol(X0) == nrow(P0)

[1] TRUE
```

The complete LEP function is,

```
R> fit <- LEP(X, y, SS = NULL, opts = NULL, logfile = "screen", lbPval = 1e-12, verbose = T)
```

The genotype data `X` and the phenotype data `y` must be specified, the remaining parameters are optional, they have default values. To be specific, `SS` is for the summary statistics, `opts` is for the runing parameter setting, `logfile` is for the log file name (the default value 'screen' indicates that the function would print the information on the screen), `lbPval` is for the restriction of the minimal value of p-values, `verbose` is for whether to print the running information. The output `fit` contains the parameters for the LEP model, the detail would be mentioned in the following part.

The parameter `opts` has two fields, '`max_iter`' for the max number of iterations and '`dis_gap`' for the display gap of the printing message. Their default values are $(600, 60)$. They could be specified individually or simultaneously by either of the following commands.

```
R> opts = list(dis_gap=1)
R> opts = list(max_iter = 300)
R> opts = list(max_iter = 300, dis_gap=1)
```

The order for the parameters does not matter.

The LEP model is fitted only with the genotype data, with no information printed:

```
R> fit <- LEP(X0, y0, opts = opts, verbose = F)
```

The LEP model integrates the genotype data {X0, y0} and summary statistics {P0} with the following command

```
R> fit <- LEP(X0, y0, SS = P0, verbose = F)
```

2.1.2 R-type data with SNPs' information

If the genotype data and summary statitistics share only part of the set of SNPs, LEP would take their intersection automaticly. The information for the genotype data and p-values is as follows.

```
R> str(X)

 num [1:1000, 1:3000] 0 0 1 1 0 0 0 0 0 0 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:3000] "rs1" "rs2" "rs3" "rs4" ...

R> str(P)

 num [1:3000, 1:3] 1.27e-05 8.61e-04 8.97e-02 2.35e-02 9.20e-01 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:3000] "rs1" "rs2" "rs3" "rs4" ...
  ..$ : chr [1:3] "lab1" "lab2" "lab3"

R> geno_snps = colnames(X)
R> ss_snps = rownames(P)
R> num_intersect <- intersect(geno_snps,ss_snps)
R> print(length(num_intersect))

[1] 2900
```

According to the above output, it could be seen that the genotype data and the summary statistics share 2900 SNPs, LEP uses the data with respect to the intersection of the SNPs to fit the model.

```
R> fit <- LEP(X, y, SS = P)
```

2.1.3 Binary plink file with R-type data storing the SNPs information

LEP package also supports the input of binary plink file, which saves huge space for the genotype data.

The complete LEP function is,

```
R> fit <- LEP_Plink(genoplinkfile, SS = NULL, opts = NULL, logfile = "screen", lbPval = 1e-12,
       verbose = F)
```

In this scene, genotype data in the plink format take the place of R-type data $\{X, y\}$

```
R> fit <- LEP_Plink(plinkfile, SS = P)
```

For the simulated data in this package, all the information contained in the plink files is the same as $\{X, y\}$ in $D2$. LEP will take intersection as it does for $D2$.

The output for the above fitting is like following

```
R> str(fit)
```

```
List of 14
 $ sigma2beta: num 0.0982
 $ sigma2e   : num 15.1
 $ gammas    : num [1:2900, 1] 0.1253 0.08287 0.00237 0.01518 0.00661 ...
 $ mu        : num [1:2900, 1] 0.096 -0.0951 0.097 0.1429 0.0738 ...
 $ S         : num [1:2900, 1] 0.0407 0.0298 0.0229 0.0245 0.0232 ...
 $ pi        : num 0.0739
 $ M         : num 2900
 $ cov       : num -1.81
 $ L         : num 698
 $ iter      : num 163
 $ u         : num [1:3, 1] 0.825 0.944 0.992
 $ v         : num [1:3, 1] 0.949 0.95 0.971
 $ fdr       : num [1:2900, 1] 0.875 0.917 0.998 0.985 0.993 ...
 $ param_beta: num [1:3, 1] 0.121 0.125 0.121
```

12 items of output are listed as above, the first 7 fields correspond to the notations $\sigma^2_\beta, \sigma^2_e, \{\gamma_j\}_1^M$, $\{\mu_j\}_1^M$, $\{s^2_j\}_1^M$, π, M. *cov* corresponds to the regression intercept for the LEP model, `L` is the final lower bound, `iter` is the total iterations taken, `u,v` are the pleitropy effect defined in the paper, `fdr` is the local false discovery rate for each variable and `param_beta` is the α parameter for each Beta distribution for the p-values.

2.2 Evaluate the performance of prediction by cross validation

This section shows how to evaluate the performance of the model in terms of prediction accuracy by cross validation. Two corresponding functions are as follows

The parameter `opts` has three fields, '`max_iter`' for the max number of iterations for each fold, '`dis_gap`' for the display gap of printing message and '`n_fold`' for the number of folds. Their default values are $(300, 60, 5)$. They could be specified individually or simultaneously by either of the following commands.

```
R> opts = list(dis_gap=1)
R> opts = list(max_iter = 300)
R> opts = list(max_iter = 300,dis_gap=1)
R> opts = list(n_fold = 5)

R> performance <- LEPCV(X, y, SS = NULL, opts = NULL, logfile = "screen",
                        lbPval = 1e-12, measure = "mse")
```

and

```
R> performance <- LEPCV_Plink(plinkfile, SS = NULL, opts = NULL, logfile = "screen",
                         lbPval = 1e-12, measure = "mse")
```

The performance could be measured by *auc* or *mse*(by default) specified by the parameter `measure`. Besides, the parameter `opts` have a field `n_fold` to specify the number of folds for cross-validation as the previous one, the default value is 5. It could be specified as

```
R> opts = list(n_fold = 10)
```

The model could be evaluated without p-values

```
R>   performance <- LEPCV(X, y)
R>   print(performance)

$mse
[1] 0.2108222
```

or with p-values

```
R>   performance <- LEPCV(X, y, SS = P,measure = "auc")
R>   print(performance)

$auc
[1] 0.807288
```

or with genotype data in the plink format

```
R> performance <- LEPCV_Plink(plinkfile, SS = P, measure = "auc")
```

2.3 Predict with the fitted model

Once a model is fitted by `LEP`, it could be used to predict the phenotype of the given genotype data by the following command.

```
R> yhat <- LEP_Predict(fit, X)
```

Please contact Mingwei Dai at `daimingwei@gmail.com` for any questions or suggestions regarding the '`LEP`' package.

References

[1] Mingwei Dai, Xiang Wan, Hao Peng, Yao Wang, Yue Liu, Jin Liu, Zongben Xu, Can Yang. LEP: Joint Analysis of Individual-level and Summary-level GWAS Data by Leveraging Pleiotropy. Submitted.