‘LPM’ Package to characterize relationship among complex traits using summary statistics from multiple GWASs and functional annotations

Jingsi Ming [1], Tao Wang [2], and Can Yang [1]

[1] Department of Mathematics, The Hong Kong University of Science and Technology, Hong Kong.
[2] Department of Bioinformatics and Biostatistics, Shanghai Jiao Tong University, Shanghai, China.

October 1, 2018

1 Overview

This vignette provides an introduction to the ‘LPM’ package. R package ‘LPM’ implements LPM (Latent Probit Model), an efficient statistical approach to characterize relationship among complex traits using summary statistics from multiple GWASs and functional annotations. It provides model parameter estimation as well as statistical inference.

The package can be loaded with the command:

```
R> library("LPM")
```

This vignette is organized as follows. Section 2.1 discusses how to fit LPM in various settings. Section 2.2 explains command lines for statistical inference for identification of risk SNPs using LPM. Section 2.3 explains command lines for the relationship test among traits. Section 2.4 explains command lines for the hypothesis testing of annotation enrichment.

Please feel free to contact Can Yang at macyang@ust.hk for any questions or suggestions regarding the ‘LPM’ package.

2 Workflow

In this vignette, we use the simulated ExampleData in the package. The number of GWASs, SNPs and functional annotations are $K = 4$, $M = 100,000$ and $D = 5$ respectively. Users can find the P-value for these GWASs in the ‘ExampleData$data’, design matrix of functional annotations in ‘ExampleData$X.

```
R> data(ExampleData)
R> length(ExampleData$data)

[1] 4

R> dim(ExampleData$data[[1]])
```

```
[1] 100000      2
```

```
R> head(ExampleData$data[[1]])
```

```
  SNP          p
1   1 0.25697603
2   2 0.42443350
3   3 0.06641065
4   4 0.24271524
5   5 0.55344380
6   6 0.28127733
```

```
R> dim(ExampleData$X)
```

```
[1] 100000      6
```

Note that our package does not require the number of SNPs for GWASs to be the same.

2.1 Fitting the LPM

We are now ready to fit LPM using the data described above (`ExampleData$data` and `ExampleData$X`). We first fit pairs of GWASs using `bLPM` and then obtain the estimates for LPM. R package `LPM` provides flexible analysis framework and automatically adjusts its model structure based on the provided data.

We can fit bLPM without or with annotation data using differenct numbers of cores with the command:

```
R> bLPMfit_noX <- bLPM(data = ExampleData$data, coreNum = 3)
```

or

```
R> bLPMfit <- bLPM(data = ExampleData$data, X = ExampleData$X, coreNum = 3)
```

Equivalently, we can first fit bLPM for the first three GWASs and then add the fourth GWAS.

```
R> bLPMfit_first3 <- bLPM(data = ExampleData$data[1:3], X = ExampleData$X, coreNum = 3)
R> bLPMfit <- bLPM_add(data = ExampleData$data[1:3], data_add = ExampleData$data[4],
+                       X = ExampleData$X, bLPMfit_first3, coreNum = 3)
```

'`bLPMfit`' is a list containing parameter estimation and the value of lower bound of log-likelihood.

```
R> str(bLPMfit)
```

```
List of 7
 $ alpha        : num [1:2, 1:6] 0.198 0.399 0.198 0.499 0.198 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:6] "P1_P2" "P1_P3" "P1_P4" "P2_P3" ...
 $ beta         : num [1:2, 1:6, 1:6] -1.007 -0.999 -1.475 0.183 0.749 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ : NULL
  .. ..$ : chr [1:6] "intercept" "X1" "X2" "X3" ...
  .. ..$ : chr [1:6] "P1_P2" "P1_P3" "P1_P4" "P2_P3" ...
 $ rho          : Named num [1:6] 0.7566 0.4364 0.5688 0.2473 0.0154 ...
  ..- attr(*, "names")= chr [1:6] "P1_P2" "P1_P3" "P1_P4" "P2_P3" ...
 $ L_stage1_List:List of 6
  ..$ P1_P2: num [1:126, 1] 30774 34710 35169 35417 35566 ...
  ..$ P1_P3: num [1:210, 1] 32979 36680 37355 37785 38084 ...
  ..$ P1_P4: num [1:60, 1] 57086 61246 63264 64427 65134 ...
  ..$ P2_P3: num [1:239, 1] 2350 8044 8374 8622 8828 ...
  ..$ P2_P4: num [1:125, 1] 26457 32611 34283 35263 35877 ...
  ..$ P3_P4: num [1:210, 1] 28663 34581 36468 37631 38395 ...
 $ L_stage2_List:List of 6
  ..$ P1_P2: num [1:893, 1] 35944 37653 38772 39518 40031 ...
  ..$ P1_P3: num [1:369, 1] 39411 41231 42443 43269 43852 ...
  ..$ P1_P4: num [1:191, 1] 66588 69438 71315 72572 73437 ...
  ..$ P2_P3: num [1:1165, 1] 10168 10501 10792 11047 11269 ...
  ..$ P2_P4: num [1:888, 1] 37345 38708 39664 40350 40854 ...
  ..$ P3_P4: num [1:368, 1] 40813 42285 43335 44102 44675 ...
 $ L_stage3_List:List of 6
  ..$ P1_P2: num [1:1391, 1] 42018 42022 42027 42032 42036 ...
  ..$ P1_P3: num [1:748, 1] 46305 46306 46306 46307 46308 ...
  ..$ P1_P4: num [1:435, 1] 76116 76127 76137 76147 76156 ...
  ..$ P2_P3: num [1:1417, 1] 13063 13063 13063 13063 13063 ...
  ..$ P2_P4: num [1:193, 1] 42874 42874 42874 42874 42874 ...
  ..$ P3_P4: num [1:89, 1] 47161 47161 47161 47161 47161 ...
 $ R            : num [1:4, 1:4] 1 0.757 0.436 0.569 0.757 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:4] "P1" "P2" "P3" "P4"
  .. ..$ : chr [1:4] "P1" "P2" "P3" "P4"
```

Then we can obtain the parameter estimation of LPM using the command:

```
R> LPMfit <- LPM(bLPMfit)

R> LPMfit

$alpha
       P1        P2        P3        P4 
0.1981940 0.3999445 0.4995458 0.3021826 

$beta
      intercept         X1         X2         X3          X4          X5
[1,] -1.0077385 -1.4720443  0.7495342  1.3064658 -1.36274367 -0.04026305
[2,] -0.9942108  0.1804042 -0.7746157 -0.3170232 -2.04512673  0.92971756
[3,] -0.9755317 -1.0420933  0.5719584  1.6872110  1.22761090  0.81497418
[4,] -1.0128372  2.1029440  0.9830136  0.5493883 -0.06902036  0.81664164

$R
          P1        P2          P3          P4
P1 1.0000000 0.7565589  0.436370769  0.568814194
```

```
P2 0.7565589 1.0000000  0.247315242  0.015424804
P3 0.4363708 0.2473152  1.000000000 -0.006535072
P4 0.5688142 0.0154248 -0.006535072  1.000000000
```

2.2 Statistical inference for identification of risk SNPs

Now, based on the fitted LPM, we can make statistical inference for identification of risk SNPs for one or more traits.

```
R> posterior1 <- post(ExampleData$data[1], X = ExampleData$X, id = 1, LPMfit)
R> posterior13 <- post(ExampleData$data[c(1, 3)], X = ExampleData$X, id = c(1, 3), LPMfit)
R> str(posterior1)

'data.frame':        100000 obs. of  2 variables:
 $ SNP      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ posterior: num  0.004705 0.00232 0.244826 0.096299 0.000316 ...

R> str(posterior13)

'data.frame':        100000 obs. of  4 variables:
 $ SNP           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ post.joint    : num  0.005058 0.001594 0.061276 0.101614 0.000204 ...
 $ post.marginal1: num  0.005073 0.00267 0.22721 0.119634 0.000236 ...
 $ post.marginal2: num  0.927 0.151 0.126 0.62 0.291 ...
```

'post' function returns list of posterior. For two GWASs, 'post.joint' means the posteriors that each SNP is associated with both the two target GWASs, 'post.marginal1' and 'post.marginal2' means the posteriors that each SNP is associated with only the first target GWAS and the second target GWAS respectively.

```
R> assoc1 <- assoc(posterior1, FDRset = 0.1, fdrControl = "global")
R> assoc13 <- assoc(posterior13, FDRset = 0.1, fdrControl = "global")
R> str(assoc1)

'data.frame':        100000 obs. of  2 variables:
 $ SNP: int  1 2 3 4 5 6 7 8 9 10 ...
 $ eta: num  0 0 0 0 0 0 1 0 0 0 ...

R> str(assoc13)

'data.frame':        100000 obs. of  4 variables:
 $ SNP          : int  1 2 3 4 5 6 7 8 9 10 ...
 $ eta.joint    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ eta.marginal1: num  0 0 0 0 0 0 1 0 0 0 ...
 $ eta.marginal2: num  1 0 0 0 0 1 1 0 0 0 ...

R> table(assoc1$eta)

    0     1 
87578 12422 
```

```
R> table(assoc13$eta.marginal1)

    0     1
87536 12464

R> table(assoc13$eta.joint)

    0     1
95415  4585
```

'`assoc`' function returns binary values indicating association of SNPs for differenct traits, where one indicates association and zero otherwise. '`assoc`' allows both local ('`fdrControl="local"`') and global FDR controls ('`fdrControl="global"`') and users can set the threshold using the argument '`FDRset`'.

2.3 Relationship test among traits

We can test the relationship among traits based on the results of pairwise analysis.

```
R> p_value_test_rho <- test_rho(bLPMfit)
R> p_value_test_rho

   P1          P2          P3        P4
P1  0 0.000000000 0.000000000 0.0000000
P2  0 0.000000000 0.000162593 0.5408885
P3  0 0.000162593 0.000000000 0.7125739
P4  0 0.540888493 0.712573921 0.0000000
```

'`test_rho`' function returns matrix of p-values for the relationship test between pairs of traits.

2.4 Hypothesis testing of annotation enrichment

When we integrate functional annotation data, we are interested in the enrichment of annotation for a specific trait. We can implement the hypothesis testing for annotation enrichment using '`test_beta`' method:

```
R> result_test_beta <- test_beta(ExampleData$data, X = ExampleData$X, id = 1, LPMfit)
R> result_test_beta

$p_value
 intercept         X1         X2         X3         X4         X5
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.06675274

$se
[1] 0.01352074 0.03848266 0.02076192 0.02189727 0.03644137 0.02196164
```

'`test_beta`' function returns the p-values of the hypothesis testing of annotation enrichment and standard errors of the estimates for the trait.

References

Jingsi Ming, Tao Wang, Can Yang; LPM: a latent probit model to characterize relationship among complex traits using summary statistics from multiple GWASs and functional annotations.