biMM vignette

Matti Pirinen* & Christian Benner
University of Helsinki

November 15, 2016

1 Introduction

biMM is a software package to efficiently estimate variance parameters of a bivariate lineax mixed model (LMM) in the following context arising in genetics research.

Consider the bivariate LMM for n individuals: $\boldsymbol{Y} = \boldsymbol{G} + \boldsymbol{\varepsilon}$, where $\boldsymbol{Y} = \left(\boldsymbol{Y}_1^T, \boldsymbol{Y}_2^T\right)^T$ is $2n$-vector of mean-centered phenotype values, $\boldsymbol{G} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_G)$ is $2n$-vector of genetic random effects and $\varepsilon \sim \mathcal{N}(0, \boldsymbol{\Sigma}_\varepsilon)$ is $2n$-vector of environmental random effects. The $(2n) \times (2n)$ covariance structures are parameterized by six parameters: genetic variances V_{G1} and V_{G2}, genetic covariance V_{G12}, environmental variances $V_{\varepsilon 1}$ and $V_{\varepsilon 2}$ and environmental covariance $V_{\varepsilon 12}$ as

$$\boldsymbol{\Sigma}_G = \left[\begin{array}{c|c} V_{G1}\boldsymbol{R} & V_{G12}\boldsymbol{R} \\ \hline V_{G12}\boldsymbol{R} & V_{G2}\boldsymbol{R} \end{array}\right] \text{ and } \boldsymbol{\Sigma}_\varepsilon = \left[\begin{array}{c|c} V_{\varepsilon 1}\boldsymbol{I} & V_{\varepsilon 12}\boldsymbol{I} \\ \hline V_{\varepsilon 12}\boldsymbol{I} & V_{\varepsilon 2}\boldsymbol{I} \end{array}\right]$$

expressed as $n \times n$ block matrices. $\boldsymbol{I}$ is the identity matrix. Typically the element i, j of the genetic realtionship matrix (GRM) $\boldsymbol{R}$ is

$$\boldsymbol{R}_{ij} = \frac{1}{K} \sum_{k=1}^{K} \left(g_{ik} - 2\widehat{f}_k\right)\left(g_{jk} - 2\widehat{f}_k\right)\left(2\widehat{f}_k\left(1 - \widehat{f}_k\right)\right)^{\alpha},$$

where g_{ik} is the genotype of individual i at variant k, coded as 0, 1 or 2 copies of the minor allele and $\widehat{f}_k$ is the minor allele frequency (MAF) and $\alpha = -1$. However, any other form for $\boldsymbol{R}$ could also be used.

An estimate of V_{Gt} approximates additive genetic variance of each trait ($t = 1, 2$) explained by the variants included in the calculation of $\boldsymbol{R}$. An estimate of the genetic correlation $\rho_G = V_{G12}/\sqrt{V_{G1}V_{G2}}$ measures (average) correlation of the allelic effects of the variants on the two traits. Similarly, we can estimate $\rho_\varepsilon = V_{\varepsilon 12}/\sqrt{V_{\varepsilon 1}V_{\varepsilon 2}}$, the correlation in the environmental components between the phenotypes.

*correspondence to matti.pirinen@helsinki.fi

2 The implementation

biMM is implemented through a *biMM.data* reference class (type "?biMM.data" in R for help). An object from that class holds the data and provides a set of functions for the analysis. This way we avoid copying the large data matrices as we operate on them using the given functions defined in the reference class. Still, sometimes you might want to manually examine the *biMM.data* in more detail, which you can do by accessing the fields

- 'traits', data.frame, phenotype values,
- 'trait.cor', matrix, Pearson correlations of traits,
- 'relatedness', matrix, GRM,
- 'IndID', vector, IDs of individuals,
- 'Included', vector, which individuals are included in current 'U' and 'd',
- 'U', matrix, eigenvectors of 'Included' individuals,
- 'd', vector, eigenvalues of 'Included' individuals.

Each field can be accessed by 'x$field' where x is a *biMM.data* object.
Install biMM package

```
install.packages("biMM_1.0.0.tar.gz", repos = NULL)
```

Load biMM package and create a new *biMM.data* object 'x'

```
library("biMM")

## Loading required package:  methods

x = biMM.data()
```

2.1 Data handling

Load genetic relationship matrix (GRM) and phenotype data for individuals from the extdata directory of the biMM package. Note that GRM comes with an .id file that contains the two columns of IDs for the individuals. The name of the .id file is always the name of GRM file extended by '.id', typically ending in '.grm.id'. Therefore we will remove the trailing '.gz' from the name of the gzipped GRM file below.

```
tr.file = system.file("extdata", "biMM_example.tr", package = "biMM")
grm.file = system.file("extdata", "biMM_example.grm.gz", package = "biMM")
grm.file = gsub(".gz$", "", grm.file, perl=T) #remove trailing .gz
x$read.traits.grm(tr.file,grm.file)
```

```
## Reading traits from biMM/extdata/biMM_example.tr
## Using ID column 2.
## Read data for 868 individuals on 4 traits.
## Reading IDs from biMM/extdata/biMM_example.grm.id
## Can't find grm file: biMM/extdata/biMM_example.grm.
##   ---> Trying .gz instead.
## Reading grm from biMM/extdata/biMM_example.grm.gz
## Read relatedness matrix for 868 individuals.

x$nInd #How many individuals do we have?

## [1] 868

x$printTraits() #What are the trait names in phenotype file?

## [1] "T1" "T2" "T3" "T4"

x$IndID[1:6] #Who are the first 6 individuals?

## [1] "IND1" "IND2" "IND3" "IND4" "IND5" "IND6"
```

Check if there are highly related pairs of individuals

```
x$relatedPairs(0.8)

## Finds related above threshold 0.8.
## Found 3 pairs of related individuals.
##     ind1   ind2         r
## 1 IND188 IND187 0.9866931
## 2 IND533 IND386 0.9902936
## 3 IND553 IND403 1.0049880
```

These three pairs might be duplicates or identical twins and we want to do some exclusions. Let' s find a minimum set of individuals to remove and then remove them:

```
exc.set = x$minimum.set.of.relateds.to.exclude(x$relatedPairs(0.8))

## Finds an approximate minimum vertex cover of relatedness graph.
## Finds related above threshold 0.8.
## Found 3 pairs of related individuals.
## Starts with 3 pairs involving 6 unique individuals.
## Suggests excluding 3 individuals.

x$excludeInd(exc.set)

## Excludes 3 individuals.
## Remaining 865 individuals.
```

For this case we ended up removing one individual from each highly related pair. In more complex cases it may, however, be possible that we can get rid of all related pairs by removing less than half of the individuals reported in those pairs, since some individuals may be members of several pairs. Therefore it is a good idea to use 'minimum.set.of.relateds.to.exclude()' to define a more optimal set of exclusions than what the naive approach might give.

Look basic stats of the traits:

```
summary(x$traits)
```

```
##        T1                  T2
##  Min.   :-3.262000   Min.   :-3.283000
##  1st Qu.:-0.626700   1st Qu.:-0.602900
##  Median :-0.009843   Median :-0.039130
##  Mean   :-0.000342   Mean   : 0.001888
##  3rd Qu.: 0.679700   3rd Qu.: 0.653350
##  Max.   : 3.204000   Max.   : 3.869000
##  NA's   :16          NA's   :14
##        T3                  T4
##  Min.   :-3.263000   Min.   :-2.897000
##  1st Qu.:-0.664350   1st Qu.:-0.706050
##  Median : 0.008850   Median :-0.053720
##  Mean   : 0.002305   Mean   :-0.000483
##  3rd Qu.: 0.687400   3rd Qu.: 0.691200
##  Max.   : 2.762000   Max.   : 3.895000
##  NA's   :22          NA's   :17
```

```
x$trait.cor #the same as cor(x$traits,method="pearson",use="pairwise")
```

```
##              T1          T2         T3           T4
## T1  1.000000000  0.51823970 -0.1173029  0.008330821
## T2  0.518239702  1.00000000 -0.1056435  0.048712393
## T3 -0.117302885 -0.10564351  1.0000000 -0.348712146
## T4  0.008330821  0.04871239 -0.3487121  1.000000000
```

If we wanted to exclude some traits (say, T3 and T4) it could be done like this:

```
#x$excludeTrait(c("T3","T4"))
```

But we have commented it out for now as in the following we consider all the traits.

2.2 Univariate analysis

Let's order the traits for univariate heritability analysis. The ordering is important to speed up the analysis if some traits are measured on completely

overlapping individuals. Additionally, assuming we tolerate some missing data, a partial overlap also speeds up the computation. Let's allow mean imputing trait values for at most 20 individuals and dropping at most 30 individuals from the analysis if by these criteria we can increase sample overlap between adjacent traits. (For multivariate normal imputation of trait values see the end of this document.)

```
tr.ord = x$order.for.heritability(trait.names=NULL,
start.trait=NULL, allow.impute=20, allow.set.missing=30)

## Found 4 traits from trait data.
## Searching for the best starting point:
##
 25 % done.
 50 % done.
 75 % done.
 100 % done.
##   first_trait new_eigen median.imputed median.ignored
## 1          T1         2           15.5           16.0
## 2          T2         2           16.0           13.5
## 3          T3         1           14.0           21.0
## 4          T4         2           14.5           16.5
## Starts from T3
```

We see that by starting from T3 we need to do only one eigendecomposition whereas by starting from any other trait we would need to do two eigendecompositions. If we would not tolerate any imputation or setting data to missing (allow.impute=0 and allow.set.missing=0) then we would need to do all possible 4 eigendecompositions. For larger data sets this would make a big difference in run time. Input parameter 'trait.names' would hold the traits included in the analysis (default NULL value considers all traits) and 'start.trait' would define the first trait in the ordering (default NULL makes biMM iterate through all possible starting points to find one leading to the smallest number of eigendecompositions). Note that if there were hundreds of traits, it may in some cases be useful to fix a start.trait in order to speed up the ordering process. Let's see how this ordering would behave with respect to eigendecompositions and missing / imputed data without doing any actual computations:

```
x$heritability(tr.ord, allow.impute=20, allow.set.missing=30,
computation=FALSE, normalize=TRUE)

##   Trait   N N_imputed N_ignored Herit SE new_eigen
## 1    T3 843         0       843    NA NA         1
## 2    T2 830        13        21    NA NA         0
## 3    T4 827        16        21    NA NA         0
## 4    T1 828        15        21    NA NA         0
```

Thus requires one eigen decomposition (for T3) and imputing at most 16 and ignoring at most 21 individuals. N is the effective number of individuals and does not count the imputed or ignored trait values. N is used for computing standard errors adjusted for the amount of observed data. Let's do the analysis:

```
x$heritability(tr.ord, allow.impute=20, allow.set.missing=30,
computation=TRUE, normalize=TRUE)

## Estimates heritability  of 4 traits.
## Trait: T3
## Would need to impute 0 and set missing 843 individuals...
## ...too much. --- Does an eigen decomposition for 843 individuals.
## Trait: T2
## Would need to impute 13 and set missing 21 individuals...
## ...OK. --- No eigen decomposition.
## Trait: T4
## Would need to impute 16 and set missing 21 individuals...
## ...OK. --- No eigen decomposition.
## Trait: T1
## Would need to impute 15 and set missing 21 individuals...
## ...OK. --- No eigen decomposition.
##   Trait   N N_imputed N_ignored     Herit         SE
## 1    T3 843         0         0 0.9104370 0.07084176
## 2    T2 830        13        21 0.6566776 0.07970500
## 3    T4 827        16        21 0.4262978 0.08569358
## 4    T1 828        15        21 0.7756641 0.07710069
##   new_eigen
## 1         1
## 2         0
## 3         0
## 4         0
```

2.3 Bivariate analysis

Let's again allow imputing trait values for at most 20 individuals and dropping at most 30 individuals from the analysis if by these criteria we can increase sample overlap between adjacent pairs of traits. To order the traits:

```
tr.ord = x$order.for.bivariate.heritability(trait.pairs=NULL,
start.pair=NULL, allow.impute=20, allow.set.missing=30)

## Found 6 trait pairs from trait data.
## Searching for the best starting point:
##
```

```
 16.7 % done.
 33.3 % done.
 50 % done.
 66.7 % done.
 83.3 % done.
 100 % done.
##   first_pair_1 first_pair_2 new_eigen median.imputed
## 1           T2           T1         2           16.0
## 2           T3           T1         1           16.0
## 3           T3           T2         1           16.0
## 4           T4           T1         2           17.5
## 5           T4           T2         2           16.0
## 6           T4           T3         1           16.0
##   median.ignored
## 1           16.0
## 2            6.5
## 3            6.5
## 4           16.0
## 5           15.5
## 6            6.5
## Starts from pair T3 -- T1
```

and to check how this would work (without doing computations) you could run

```
x$bivariate.heritability(tr.ord, allow.impute=20,
   allow.set.missing=30, computation=FALSE, normalize=TRUE)
```

Let's do the computations

```
x$bivariate.heritability(tr.ord , allow.impute=20,
   allow.set.missing=30, computation=TRUE, normalize=TRUE)

## Estimates bivariate model of 6 pairs.Traits: T3 T1
## Would need to impute for 15 individuals and set missing values
## for 0 individuals ...OK. --- No eigen decomposition.
## Traits: T3 T2
## Would need to impute for 13 individuals and set missing values
## for 0 individuals ... OK. --- No eigen decomposition.
## Traits: T4 T3
## Would need to impute for 16 individuals and set missing values
## for 0 individuals ... OK. --- No eigen decomposition.
## Traits: T4 T2
## Would need to impute for 28 individuals and set missing values
## for 20 individuals ... too much. --- Does an eigen decomposition
```

```
## for 835 individuals.
## Traits: T4 T1
## Would need to impute for 16 individuals and set missing values
## for 13 individuals ... OK. --- No eigen decomposition.
## Traits: T2 T1
## Would need to impute for 16 individuals and set missing values
## for 16 individuals ... OK. --- No eigen decomposition.
## Optimized with BFGS: 6  NM: 0
##   Trait1 Trait2   N N_imputed N_ignored       Vg1
## 1     T3     T1 828        15         0 0.8934000
## 2     T3     T2 830        13         0 0.8714523
## 3     T4     T3 827        16         0 0.4242683
## 4     T4     T2 835         0         0 0.4713549
## 5     T4     T1 819        16        13 0.4591180
## 6     T2     T1 819        16        16 0.6396743
##       Vg1_SE        Ve1     Ve1_SE       Vg2     Vg2_SE
## 1 0.10462581 0.09972146 0.06900432 0.7684915 0.10104619
## 2 0.10393556 0.12443820 0.06963678 0.6472959 0.09620165
## 3 0.09119358 0.55802425 0.08092243 0.8813337 0.10345914
## 4 0.09495813 0.52964710 0.08237684 0.6650321 0.09630476
## 5 0.09451116 0.52297189 0.08214701 0.7968151 0.10132537
## 6 0.09480605 0.33956306 0.07306623 0.7959815 0.10123264
##         Ve2     Ve2_SE       cor_g   cor_g_SE        cor_e
## 1 0.2196151 0.07196900 -0.25286604 0.08501530  0.608416692
## 2 0.3382069 0.07398375 -0.15925638 0.09250376  0.065862495
## 3 0.1093856 0.06863228 -0.59193666 0.09117581  0.062998792
## 4 0.3330331 0.07369740  0.02828162 0.12034702  0.078552741
## 5 0.1869276 0.07088431  0.02345524 0.11385676 -0.006628343
## 6 0.1875906 0.07086677  0.64040829 0.06422118  0.212227982
##    cor_e_SE         cor     cor_SE new_eigen
## 1 0.4394412 -0.11730288 0.03424470         0
## 2 0.2512733 -0.10564351 0.03428980         0
## 3 0.2311822 -0.34871215 0.03231827         0
## 4 0.1305989  0.04871239 0.03460681         1
## 5 0.1712676  0.01295082 0.03464504         0
## 6 0.1876874  0.51964323 0.02960263         0
```

Note that there is only one eigendecomposition done (for pair T4–T1). In particular, the bivariate analysis made use of the eigendecomposition that was currently stored in the *biMM.data* object by starting the bivariate analyses from the pairs that were highly overlapping with the current set of 'Included' individuals.

2.4 Misc.

If you want to save the current *biMM.data* object, for example, because it holds an eigendecomposition for a large data set, you should use the standard 'saveRDS' function of R and then later read it in using 'readRDS'. There are no specific output-to-file functions written for biMM, because biMM outputs the results as data.frames and you can write those to a file with the standard 'write.table' command.

Trait value imputation is done as an expected value from the multivariate normal distribution given the individual's values on (at most) k observed traits that are most highly correlated (in absolute value) with the target trait. Parameter k is set by 'traits.for.imputation' in functions 'heritability' and 'bivariate.heritability' and defaults to 0. If $k = 0$ then the mean imputation is used, i.e., the imputed value is the mean of the observed trait values among the other individuals. If you set $k > 0$ make sure that the other traits in the 'traits' are such that you truly want to use them for imputing the traits of interest. If they are not, then you should first make exclusions to 'traits'.

2.5 Input file formats

Traits. An ASCII file where the first two columns are individuals' IDs (e.g. FAM_ID and IND_ID in PLINK's PED files[1]). The remaining columns are read in as traits, but those not of interest can then be excluded. All non-numeric trait columns are removed. There should be a header line naming all the columns. Missing values should be 'NA'.
GRM. biMM reads GRM using the format of GCTA[2]. That is, there are two files grmfile.grm (can also be in zipped form grmfile.grm.gz) for the actual relationship matrix and grmfile.id for individuals' indexes. grmfile.grm has four columns for each pair of individuals (including the pairing of an individual with itself)

```
ind1 ind2 m r
```

First two columns are numerical indexes of the individuals with ind2 $\leq$ ind1, m is the number of variants used in computing the relatedness value r. biMM ignores m. There is no required ordering of pairs. If some pairs are missing from the file their relatedness value is set to 0 and a warning about this is printed out.
IDs. IDs in grmfile.id and trait file must match exactly with each other. This is to make sure that we are working with the correct data. When reading in the data, the user can choose which one of the columns 1 or 2 will be used as IndID in *biMM.data* object by setting the 'id.col' parameter to 1 or 2. The column chosen must have unique IDs for all individuals. If neither column has unique IDs it is possible to merge the two columns to form IndIDs with '_' acting as the separator by setting 'merge.ids=TRUE'.

[1] http://pngu.mgh.harvard.edu/~purcell/plink/
[2] http://cnsgenomics.com/software/gcta/

It is possible to read in only one of the files at a time, for example, to study either the traits or the relatedness structure alone.