GMMAT: Generalized linear Mixed Model Association Tests
Version 0.9.2

Han Chen
Human Genetics Center
Dept. of Epidemiology, Human Genetics and Environmental Sciences
Center for Precision Health
School of Public Health & School of Biomedical Informatics
The University of Texas Health Science Center at Houston
Email: Han.Chen.2@uth.tmc.edu

June 8, 2018

Contents

1 Introduction

GMMAT is an R package for performing association tests using generalized linear mixed models (GLMMs)[1] in genome-wide association studies (GWAS) and sequencing association studies. GLMMs provide a broad range of models for correlated data analysis. In the GWAS and sequencing association study context, examples of correlated data include those from family studies, samples with cryptic relatedness and/or shared environmental effects, as well as samples generated from complex sampling designs.

GMMAT first fits a GLMM with covariate adjustment and random effects to account for population structure and family or cryptic relatedness. For GWAS, *GMMAT* performs score tests for each genetic variant. For candidate gene studies, *GMMAT* can also perform Wald tests to get the effect size estimate for each genetic variant. For rare variant analysis from sequencing association studies, *GMMAT* performs the burden test,[2–5] the sequence kernel association test (SKAT),[6] SKAT-O[7] and the efficient hybrid test of the burden test and SKAT, in the variant Set Mixed Model Association Tests (SMMAT, Chen *et al.* 2018, manuscript in preparation) framework, based on user-defined variant sets.

2 The model

In the context of single variant test, *GMMAT* works with the following GLMM

$$\eta_i = g(\mu_i) = \mathbf{X}_i\boldsymbol{\alpha} + G_i\beta + b_i.$$

We assume that given the random effects $\mathbf{b}$, the outcome y_i are conditionally independent with mean $E(y_i|\mathbf{b}) = \mu_i$ and variance $Var(y_i|\mathbf{b}) = \phi a_i^{-1} v(\mu_i)$, where ϕ is the dispersion parameter (for binary and Poisson data $\phi = 1$), a_i are known weights, and $v(\cdot)$ is the variance function. The linear predictor η_i is a monotonous function of the conditional mean μ_i via the link function $\eta_i = g(\mu_i)$. $\mathbf{X}_i$ is a $1 \times p$ row vector of covariates for subject i, $\boldsymbol{\alpha}$ is a $p \times 1$ column vector of fixed covariate effects including the intercept, G_i is the genotype of the genetic variant of interest for subject i, and β is the fixed genotype effect. We assume that $\mathbf{b} \sim N(0, \sum_{k=1}^{K} \tau_k \mathbf{V}_k)$ is an $n \times 1$ column vector of random effects, τ_k are the variance component parameters, $\mathbf{V}_k$ are known $n \times n$ matrices. In practice, $\mathbf{V}_k$ can be the theoretical kinship matrix if analyzing family samples with known pedigree structure in a homogeneous population, or the empirical genetic relationship matrix (GRM) to account for population structure and cryptic relatedness, or any $n \times n$ matrices to account for shared environmental effects or complex sampling designs.

GMMAT can be used to analyze both continuous and binary traits. For continuous traits, if a normal distribution and an identity link function are assumed, *GMMAT* performs association tests based on linear mixed models (LMMs). For binary traits, however, we showed that performing association tests based on LMMs can lead to invalid P values in the presence of moderate or strong population stratification, even after adjusting for top ancestry principal components (PCs) as fixed effects.[8] In such scenarios, we would recommend assuming a Bernoulli distribution and a logit link function for binary traits, adjusting for top ancestry PCs as fixed-effect covariates. This GLMM is also known as the logistic mixed model.

In the context of variant set tests (also known as gene-based tests or aggregate variant tests), *GMMAT* works with the following GLMM

$$\eta_i = g(\mu_i) = \mathbf{X}_i\boldsymbol{\alpha} + \mathbf{G}_i\boldsymbol{\beta} + b_i,$$

where $\mathbf{G}_i$ is now a $1 \times q$ row vector of genotypes for q variants in subject i, $\boldsymbol{\beta}$ is a $q \times 1$ column vector of genotype effects with meanβ_0 and variance σ^2. The following 4 tests in the SMMAT framework are implemented. The burden test is for $H_0 : \beta_0 = 0$ versus $H_1 : \beta_0 \neq 0$ under the constraint $\sigma^2 = 0$, SKAT is for $H_0 : \sigma^2 = 0$ versus $H_1 : \sigma^2 > 0$ under the constraint $\beta_0 = 0$, SKAT-O is a linear combination of burden test and SKAT statistics, and the efficient hybrid test combines the burden test with an adjusted SKAT (which is asymptotically independent with the burden test) $H_0 : \sigma^2 = 0$ versus $H_1 : \sigma^2 > 0$ under no constraints on β_0 using Fisher's method. Of note, in all these tests, the null GLMM is the same as the null model in the single variant test, which only needs to be fitted once.

3 Getting started

3.1 Downloading *GMMAT*

GMMAT can be downloaded at `https://github.com/hanchenphd/GMMAT`. It can be installed as a regular R package on a UNIX operating system. Currently we do not support Mac or Windows versions.

3.2 Installing *GMMAT*

Before installing *GMMAT*, please check your system to make sure C++ libraries boost and Armadillo have been installed appropriately. *GMMAT* links to R packages Rcpp and RcppArmadillo, and also imports R packages Rcpp, CompQuadForm, foreach, doMC, parallel, Bioconductor packages SeqArray and SeqVarTools. These dependencies should be installed before installing *GMMAT*.

For optimal computational performance, it is recommended to use an R version configured with the Intel Math Kernel Library (or other fast BLAS/LAPACK libraries). See the instructions on building R use with Intel MKL (`https://software.intel.com/en-us/articles/using-intel-mkl-with-r`).

Here is an example for installing *GMMAT* and all its dependencies in an R session (assuming none of the R packages other than the default has been installed):

```
> ## try http:// if https:// URLs are not supported
> install.packages(c("devtools", "RcppArmadillo", "CompQuadForm", "doMC",
+         "foreach"), repos = "https://cran.r-project.org/")
> source("https://bioconductor.org/biocLite.R")
> biocLite(c("SeqArray", "SeqVarTools"))
> devtools::install_github("hanchenphd/GMMAT")
```

3.3 Using *GMMAT* in Analysis Commons

The current version of the *GMMAT* package (version 0.9.1) is also available in Analysis Commons[9] on DNAnexus cloud computing platform (`https://platform.dnanexus.com/`), as the "gmmat_v0.9.1" App in Project "Commons_Tools".

4 Input

GMMAT requires the phenotype and covariates in an R data frame, known $n \times n$ matrices $\mathbf{V}_k$ as an R matrix (in the case of single matrix) or an R list (in the case of multiple matrices), and genotypes saved in a plain text file (or in a compressed plain text file .gz or .bz2), a PLINK binary PED file, or in a GDS format file. We describe how to prepare these data below.

4.1 Phenotype and covariates

Phenotype and covariates should be either saved as a data frame in R, or recorded in a text file that can be read into R as a data frame. The rows of the data frame represent different individuals, and the columns represent different variables. For example, here we show the header and first 6 rows of the example text file pheno.txt:

```
disease trait   age     sex
1       5.45    61      0
1       5.61    50      1
0       3.1     54      0
1       6.22    48      1
1       5.42    49      0
0       6.22    50      1
...
```

In this example, there are one binary phenotype (disease), one quantitative phenotype (trait) and two covariates (age and sex). There can be additional columns for unused variables, and the order of columns does not matter. To read it into R as a data frame, you can use

```
> pheno.file <- system.file("extdata", "pheno.txt", package = "GMMAT")
> pheno <- read.table(pheno.file, header = TRUE)
```

Missing values in the data frame should be recognizable by R as NA. For example, if you use . (period) to denote missing values in the text file, you can use

```
> pheno <- read.table(pheno.file, header = TRUE, na.strings = ".")
```

4.2 Matrices of covariance structure

GMMAT requires at least one $n \times n$ matrix $\mathbf{V}_k$ to model the covariance structure of the random effects. In the simplest case, this is usually a GRM estimated from the genotype data. Currently *GMMAT* does not provide a function to calculate the GRM, but there are many software packages that can do this job. For example, GEMMA[10] can be used to estimate either the centered GRM or the standardized GRM. GRM saved in an external file must be read into R as a matrix. For example,

```
> GRM.file <- system.file("extdata", "GRM.txt.bz2", package = "GMMAT")
> GRM <- as.matrix(read.table(GRM.file))
> ## scan is much faster than read.table if the matrix size is known:
> ## GRM <- matrix(scan(GRM.file, n = 400^2), nrow = 400, ncol = 400,
> ##         byrow = TRUE)
```

Multiple $n \times n$ matrices can be used to allow multiple components of random effects. In such cases, the matrices should be constructed as a list of $n \times n$ matrices. For example, if you have 3 R matrices Mat1, Mat2 and Mat3, you can construct the R list

```
> Mats <- list(Mat1, Mat2, Mat3)
```

All matrices must be positive semi-definite. Before running the analysis, it is very important to make sure that the dimensions of these matrices must match the row number of the phenotype data frame, and the order of individuals in the rows and columns of these matrices must also match the order of individuals in the phenotype data frame. If not, it is usually easier to sort the phenotype data frame to match the order of individuals in the matrices.

4.3 Genotypes

GMMAT can take genotype files either in plain text format (or the compressed version .gz or .bz2), PLINK binary PED format, or in the GDS format. Non-integer imputed genotypes (dosages) should be saved in plain text files (or the compressed version .gz or .bz2). The plain text file can be space-, tab-, comma-, or even special character-delimited, and there can be additional rows (e.g., comments) and/or columns before the genotype data matrix. Here is an example of part of a tab-delimited plain text genotype file geno.txt:

```
# This is an example genotype file for demonstrating GMMAT
# Each row represents one SNP for all individuals in the study
# First column is SNP name, second and third columns are alleles (Allele1
and Allele2): it is recommended to use Allele1 for the reference allele
and Allele2 for the effect allele, but reversed coding is also allowed
and does not affect association test results (users should be cautious
with allele coding when interpreting results)
# Starting from fourth column, each column represents one individual
# In this example, there are 400 individuals and 100 SNPs
SNP1    A       T       0       0       NA      NA      ...
SNP2    A       C       1       0       1       0       ...
SNP3    A       C       0       0       0       1       ...
SNP4    A       G       1       0       1       1       ...
SNP5    A       G       1       0       2       1       ...
...
```

Genotypes in Variant Call Format (VCF) and PLINK binary PED format can be converted to the GDS format using seqVCF2GDS and seqBED2GDS functions from the SeqArray package:

```
> SeqArray::seqVCF2GDS("VCF_file_name", "GDS_file_name")
> SeqArray::seqBED2GDS("BED_file_name", "FAM_file_name", "BIM_file_name",
+         "GDS_file_name")
```

5 Running *GMMAT*

If *GMMAT* has been successfully installed, you can load it in an R session using

```
> library(GMMAT)
```

We provide 6 functions in *GMMAT*: **glmmkin** for fitting the GLMM with known $\mathbf{V}_k$, **glmm.score** for running score tests, **glmm.wald** for Wald tests, **glmm.score.meta** for performing meta-analysis on score test results, **glmm.rvtests** for running variant set tests (also known as gene-based tests or aggregate variant tests), and **glmm.rvtests.meta** for performing variant set tests meta-analysis using intermediate files (single variant scores and their covariance matrices in each variant set). Details about how to use these functions, their arguments and returned values can be found in the R help document of *GMMAT*. For example, to learn more about **glmmkin**, in an R session you can type

```
> ?glmmkin
```

5.1 Fitting GLMM

Here we provide a simple example of fitting GLMM using **glmmkin**. We have the binary phenotype disease, the quantitative phenotype trait, and two covariates age and sex, saved in a plain text file pheno.txt. We also have computed the GRM externally and saved it in a compressed file GRM.txt.bz2, and have checked to make sure the order of individuals in the GRM matches the order of individuals in the phenotype data (VERY IMPORTANT). In this example we fit a GLMM assuming Bernoulli distribution of the disease and logit link function (also known as a logistic mixed model). We adjust for age and sex, and use one $n \times n$ matrix as $\mathbf{V}_k$ (the GRM) to model the covariance structure of the random effects.

```
> model0 <- glmmkin(disease ~ age + sex, data = pheno, kins = GRM,
+          family = binomial(link = "logit"))
> model0$theta

[1] 1.0000000 0.3377331

> model0$coefficients

[1]  0.472081177 -0.006818634 -0.086444746

> model0$cov

             [,1]          [,2]         [,3]
[1,]  1.21381795 -0.0228770376 -0.041621900
[2,] -0.02287704  0.0004506975  0.000393544
[3,] -0.04162190  0.0003935440  0.043649530
```

Note that pheno and GRM must be read into R as a data frame and a matrix, respectively. When using the function **glmmkin**, the data frame of phenotype and covariates (in our example, pheno) should be passed to the argument "data", and the matrix or the list of matrices for random effects (in our example, the matrix GRM) should be passed to the argument "kins". The first argument of the function **glmmkin** is "fixed", which requires a formula for fixed effects. The syntax of the formula is the same as the formula used in a linear model **lm** and a generalized linear model **glm**. The example model above is equivalent to

```
> model0 <- glmmkin(fixed = disease ~ age + sex, data = pheno,
+         kins = GRM, family = binomial(link = "logit"))
```

The argument "family" takes the same syntax as used in a generalized linear model **glm**. For example, if you would like to fit a LMM for a quantitative trait, you can use

```
> model1 <- glmmkin(fixed = trait ~ age + sex, data = pheno,
+         kins = GRM, family = gaussian(link = "identity"))
```

Please avoid using LMMs for binary traits.

To fit a heteroscedastic LMM for a quantitative trait (allowing heterogeneous residual variances among different groups),[11] you can use

```
> model2 <- glmmkin(fixed = trait ~ age + sex, data = pheno, kins = GRM,
+         groups = "disease", family = gaussian(link = "identity"))
> model2$theta

[1] 0.9082505 1.0589986 1.0853532
```

In this example, groups are defined by disease status. Therefore, disease cases and controls are assumed to have different residual variances on the quantitative trait, after adjusting for age and sex.

Here is a list of supported **family** objects (for details and alternative link/variance functions, please see the R help document of **family**):

Family	Link	Trait	Variance
binomial	logit	binary	$\mu(1-\mu)$
gaussian	identity	continuous	ϕ
Gamma	inverse	continuous	$\phi\mu^2$
inverse.gaussian	1/mu^2	continuous	$\phi\mu^3$
poisson	log	count	μ
quasi	identity	continuous	ϕ
quasibinomial	logit	binary	$\phi\mu(1-\mu)$
quasipoisson	log	count	$\phi\mu$

The function **glmmkin** returns a list. Except for the heteroscedastic LMM, the first element in the vector theta is the estimate of the dispersion parameter ϕ (for binary and Poisson data we have a fixed $\phi = 1$), and the remaining elements are variance component estimates for each matrix modeling the covariance structure of the random effects (in the same order as in the list of matrices passed to "kins"). In our binary disease example model0 above, we have only one such matrix (the GRM), and the results show that the estimate of the variance component parameter τ_k is 0.337731. The vector coefficients gives the fixed effect estimates, and the order matches the order of covariates in the formula passed to "fixed". In our binary disease example above, we have an intercept 0.472081177, and an age effect estimate -0.006818634, a sex effect estimate -0.086444746. The matrix cov is the covariance matrix of the fixed effect estimates.

When a heteroscedastic LMM is fitted using the function **glmmkin**, the first elements (with length equal to the number of groups) in the vector theta are the estimated residual variances for each group (in the same order as the levels of groups appearing in the data), followed by variance component estimates for each matrix modeling the covariance

structure of the random effects (in the same order as in the list of matrices passed to "kins"). In our heteroscedastic LMM model2 above, 0.9082505 is the residual variance estimate for cases, 1.0589986 is the residual variance estimate for controls (since in the data pheno, disease 1 appears before disease 0), and 1.0853532 is the estimate of the variance component parameter for the GRM.

In the following example, we have the same binary phenotype disease, covariates age and sex, and the same GRM as in the previous example. In addition to the GRM, we have another $n \times n$ matrix to model the covariance structure of the random effects. The GLMM is

$$\log(\frac{P(disease_i = 1|age_i, sex_i, b_i)}{1 - P(disease_i = 1|age_i, sex_i, b_i)}) = \alpha_0 + \alpha_1 \times age_i + \alpha_2 \times sex_i + b_i,$$

where $\mathbf{b} \sim N(0, \tau_1\mathbf{V}_1 + \tau_2\mathbf{V}_2)$, $\mathbf{V}_1$ is the GRM, $\mathbf{V}_2$ is a block diagonal matrix with block size 10 and all entries equal to 1 within a block. Here $\mathbf{V}_2$ is used to model clusters: in this example we have 40 clusters with 10 individuals in each cluster. For individual i in cluster j ($j = 1, 2, \ldots, 40$), the model above is equivalent to

$$\log(\frac{P(disease_i = 1|age_i, sex_i, b_{1i}, b_{2j})}{1 - P(disease_i = 1|age_i, sex_i, b_{1i}, b_{2j})}) = \alpha_0 + \alpha_1 \times age_i + \alpha_2 \times sex_i + b_{1i} + b_{2j},$$

where $\mathbf{b}_1 \sim N(0, \tau_1\mathbf{V}_1)$, $b_{2j} \sim N(0, \tau_2)$. All 10 individuals in a cluster share a common b_{2j}, and b_{2j} are independent and identically distributed across clusters.

```
> M10 <- matrix(0, 400, 400)
> for(i in 1:40) M10[(i-1)*10+(1:10), (i-1)*10+(1:10)] <- 1
> Mats <- list(GRM, M10)
> model3 <- glmmkin(fixed = disease ~ age + sex, data = pheno,
+         kins = Mats, family = binomial(link = "logit"))
> model3$theta

[1] 1.0000000 0.2199086 0.1219592
```

In this example, the dispersion parameter ϕ is fixed to 1, the variance component estimates $\hat{\tau}_1 = 0.2199086$, $\hat{\tau}_2 = 0.1219592$, corresponding to the $n \times n$ matrices $\mathbf{V}_1$ (the GRM) and $\mathbf{V}_2$ (the block diagonal matrix M10), respectively. Note that the order of individuals in $\mathbf{V}_1$ and $\mathbf{V}_2$ must match, and should also match the order of individuals in the phenotype data frame.

5.2 Single variant tests

5.2.1 Score tests

When performing score tests in GWAS, we need a fitted GLMM under the null hypothesis $H_0 : \beta = 0$ and a genotype file. We can construct score tests using the glmmkin class object returned from the function **glmmkin** for the null GLMM. Note that score tests require only vector/matrix multiplications and are much faster than Wald tests, which require fitting a new GLMM for each SNP. Score tests give the direction of effects but not effect size estimates. However, we can simply add score statistics and their variances from different studies to perform a meta-analysis. Here we provide a simple example of score tests using the plain text genotype file "geno.txt":

```
> geno.file <- system.file("extdata", "geno.txt", package = "GMMAT")
> glmm.score(model0, infile = geno.file, outfile =
+         "glmm.score.text.testoutfile.txt", infile.nrow.skip = 5,
+         infile.ncol.skip = 3, infile.ncol.print = 1:3,
+         infile.header.print = c("SNP", "Allele1", "Allele2"))
```

```
[1] "Computational time: 0.29 seconds"
```

The first argument in **glmm.score** is the returned glmmkin class object from the null GLMM. The argument "infile" is the name (and path if not in the current working directory) of the plain text genotype file (or compressed files .gz and .bz2), and the argument "outfile" is the name of the output file. In this example genotype file, we have 5 comment lines to skip using "infile.nrow.skip". The first 3 columns contain information on SNP name and alleles, which we skip from the analysis using "infile.ncol.skip" but subsequently keep in the output file using "infile.ncol.print" to select the 1st, 2nd and 3rd columns. Corresponding column names in the output file can be assigned using "infile.header.print".

If your genotype information is saved in a PLINK binary PED file "geno.bed", you can use:

```
> geno.file <- strsplit(system.file("extdata", "geno.bed",
+         package = "GMMAT"), ".bed", fixed = TRUE)[[1]]
> glmm.score(model0, infile = geno.file, outfile =
+         "glmm.score.bed.testoutfile.txt")
```

```
[1] "Computational time: 0.19 seconds"
```

Here "infile" is the prefix (and path if not in the current working directory) of the PLINK files (.bed, .bim and .fam). SNP information in the .bim file (in our example, "geno.bim") is carried over to the output file.

Alternatively, if your genotype information is saved in a GDS file "geno.gds", you can use:

```
> geno.file <- system.file("extdata", "geno.gds", package = "GMMAT")
> glmm.score(model0, infile = geno.file, outfile =
+         "glmm.score.gds.testoutfile.txt")
```

```
# of selected samples: 400
# of selected variants: 100
# of selected variants: 100
```

The function **glmm.score** returns the actual computation time in seconds from its function call for plain text genotype files (or compressed files .gz and .bz2) and PLINK binary PED files. For GDS genotype files, no value is returned.

5.2.2 Wald tests

When performing Wald tests for candidate SNPs to get effect size estimates, we need the phenotype (and covariates) data frame, the matrices modeling the covariance structure of the random effects, and the genotype file. To perform Wald tests, we do not need fitting the null GLMM required in score tests using **glmmkin**. In the example below, we perform Wald tests for 4 candidate SNPs of interest and get their effect estimates:

```
> geno.file <- system.file("extdata", "geno.txt", package = "GMMAT")
> snps <- c("SNP10", "SNP25", "SNP1", "SNP0")
> glmm.wald(fixed = disease ~ age + sex, data = pheno, kins = GRM,
+         family = binomial(link = "logit"), infile = geno.file,
+         snps = snps, infile.nrow.skip = 5, infile.ncol.skip = 3,
+         infile.ncol.print = 1:3, infile.header.print =
+         c("SNP", "Allele1", "Allele2"))
```

```
    SNP Allele1 Allele2   N         AF       BETA        SE      PVAL converged
1 SNP10       A       G 400 0.23250000 -0.1397665 0.1740090 0.4218510      TRUE
2 SNP25       A       C 400 0.17500000 -0.0292076 0.1934447 0.8799861      TRUE
3  SNP1       A       T 393 0.02544529  0.4566063 0.4909946 0.3523907      TRUE
4  SNP0    <NA>    <NA>  NA         NA         NA        NA        NA        NA
```

The syntax is a hybrid of **glmmkin** and **glmm.score**. Note that the argument "fixed" is a formula including the covariates but NOT the test SNPs. The argument "snps" is a character vector of the names of the test SNPs. If "infile" is a plain text genotype file (or compressed files .gz and .bz2), the function **glmm.wald** returns a data frame with first columns copied from the genotype file using "infile.ncol.print" and names specified using "infile.header.print", followed by the sample size N, the allele frequency (AF) of the effect allele (Allele2 in this example, but you can also define Allele1 as the effect allele in your coded genotype file), effect size estimate BETA of the effect allele, standard error SE, Wald test P value PVAL, and an indicator for whether the GLMM is converged. Note that in the example above, SNP0 is not actually included in the genotype file, so all results are missing.

If your genotype information is saved in a PLINK binary PED file "geno.bed", you can use:

```
> geno.file <- strsplit(system.file("extdata", "geno.bed",
+         package = "GMMAT"), ".bed", fixed = TRUE)[[1]]
> glmm.wald(fixed = disease ~ age + sex, data = pheno, kins = GRM,
+         family = binomial(link = "logit"), infile = geno.file,
+         snps = snps)
```

```
   CHR   SNP   cM  POS   A1   A2   N         AF       BETA        SE      PVAL
1    1 SNP10    0   10    G    A 400 0.7675000  0.1397665 0.1740090 0.4218510
2    1 SNP25    0   25    C    A 400 0.8250000  0.0292076 0.1934447 0.8799861
3    1  SNP1    0    1    T    A 393 0.9745547 -0.4566063 0.4909946 0.3523907
4 <NA>  SNP0 <NA> <NA> <NA> <NA>  NA         NA         NA        NA        NA
  converged
1      TRUE
2      TRUE
3      TRUE
4        NA
```

It returns a data frame with first 6 columns copied from the .bim file (in our example, "geno.bim"), followed by the sample size N, the allele frequency (AF) of A2 allele (the effect allele, note that A1 allele in .bim is coded 0 and A2 allele is coded 1), effect size estimate BETA of A2 allele, standard error SE, Wald test P value PVAL, and an indicator for whether the GLMM is converged.

Alternatively, if your genotype information is saved in a GDS file "geno.gds", you can use:

```
> geno.file <- system.file("extdata", "geno.gds", package = "GMMAT")
> glmm.wald(fixed = disease ~ age + sex, data = pheno, kins = GRM,
+         family = binomial(link = "logit"), infile = geno.file,
+         snps = snps)

    SNP  CHR  POS  REF  ALT   N        AF       BETA        SE      PVAL
1 SNP10    1   10    G    A 400 0.7675000  0.1397665 0.1740090 0.4218510
2 SNP25    1   25    C    A 400 0.8250000  0.0292076 0.1934447 0.8799861
3  SNP1    1    1    T    A 393 0.9745547 -0.4566063 0.4909946 0.3523907
4  SNP0 <NA> <NA> <NA> <NA>  NA        NA         NA        NA        NA
  converged
1      TRUE
2      TRUE
3      TRUE
4        NA
```

It returns a data frame with first 5 columns information extracted from the GDS file, followed by the sample size N, the allele frequency (AF) of ALT allele, effect size estimate BETA of ALT allele, standard error SE, Wald test P value PVAL, and an indicator for whether the GLMM is converged.

5.2.3 Meta-analysis

Score test results from multiple studies can be combined in meta-analysis. We provide the function **glmm.score.meta** to perform meta-analysis on score test results. Generally, if each study performs score tests using genotypes in PLINK binary PED format or GDS format, the score test output from **glmm.score** can be directly used as input files. Otherwise the meta-analysis function needs a tab or space delimited plain text file (or compressed files that can be recognized by the R function **read.table**) with at least 8 columns: SNP name, effect allele, reference allele, N, AF, SCORE, VAR and PVAL. Note that the SNP name, effect allele, reference allele can have customized column names in different input files, but the column names of N, AF, SCORE, VAR and PVAL should match exactly. Customized SNP and alleles column names can be specified using "SNP", "A1" and "A2". Note that we do not define whether "A1" or "A2" is the effect allele: it is your choice. However, your choice should be consistent across different studies: for example, if you have two studies with the same allele column names "Allele1" and "Allele2", and you want to define "A1" as the effect allele, but the effect allele is "Allele1" in the first study and "Allele2" in the second study, you need A2 = c("Allele2", "Allele1") for the reference allele column in each study, and A1 = c("Allele1", "Allele2") for the effect allele column in each study. Note that in **glmm.score** output from analyzing PLINK binary PED format genotypes, the effect allele has column name "A2", and in **glmm.score** output from analyzing GDS format genotypes, the effect allele has column name "ALT". Thus if you have a result file from analyzing PLINK binary PED format genotypes in the third study, and another result file from analyzing GDS format genotypes in the fourth study, in addition to the aforementioned two studies, and you still want to define "A1" as the effect allele, you need A2 = c("Allele2", "Allele1", "A1", "REF") for the

reference allele column in each study, and A1 = c("Allele1", "Allele2", "A2", "ALT") for the effect allele column in each study.

Here is an example of meta-analyzing 3 score test result files:

```
> meta1.file <- system.file("extdata", "meta1.txt", package = "GMMAT")
> meta2.file <- system.file("extdata", "meta2.txt", package = "GMMAT")
> meta3.file <- system.file("extdata", "meta3.txt", package = "GMMAT")
> glmm.score.meta(files = c(meta1.file, meta2.file, meta3.file),
+         outfile = "glmm.score.meta.testoutfile.txt",
+         SNP = rep("SNP", 3), A1 = rep("A1", 3), A2 = rep("A2", 3))

The following SNPs have been removed due to inconsistent alleles across studies:
[1] "L10" "L12" "L15"
```

5.3 Variant set tests

5.3.1 Pooled analysis

Variant set tests (also known as gene-based tests or aggregate variant tests) in a single study (or a pooled analysis of multiple studies) can be performed using the function **glmm.rvtests**. Currently only the GDS genotype format is supported. In addition to a glmmkin class object returned from the function **glmmkin** for the null GLMM and the GDS format genotype file, a group definition file with no header and 6 columns (variant set id, variant chromosome, variant position, variant reference allele, variant alternate allele, weight) is also required. For example, here we show the first 6 rows of the example group definition file "SetID.withweights.txt":

```
Set1    1       1       T       A       1
Set1    1       2       A       C       4
Set1    1       3       C       A       3
Set1    1       4       G       A       6
Set1    1       5       A       G       9
Set1    1       6       C       A       9
```

Note that each variant in the group definition file is matched by chromosome, position, reference allele and alternate allele with variants from the GDS file. One genetic variant can be included in different groups with possibly different weights. If no external weights are needed in the analysis, simply replace the 6th column by all 1's.

Four variant set tests are supported in the SMMAT framework: "B" for the burden test, "S" for SKAT, "O" for SKAT-O and "E" for the efficient hybrid test of the burden test and SKAT. You can include one or more tests in a single analysis. If "O" is selected, the burden test and SKAT results will be automatically included; if "E" is selected, the burden test results will be automatically included. Therefore, the following example gives all four test results:

```
> group.file <- system.file("extdata", "SetID.withweights.txt",
+         package = "GMMAT")
> geno.file <- system.file("extdata", "geno.gds", package = "GMMAT")
> glmm.rvtests(model0, group.file = group.file, geno.file = geno.file,
+         MAF.range = c(1e-7, 0.5), miss.cutoff = 1, method = "davies",
+         tests = c("O", "E"))
```

```
# of selected samples: 400
  group n.variants miss.min miss.mean miss.max freq.min freq.mean freq.max
1  Set1         20        0  0.000875   0.0175   0.5000 0.8150402  0.99125
2  Set2         20        0  0.000000   0.0000   0.6400 0.8795625  0.99125
3  Set3         20        0  0.000000   0.0000   0.5675 0.8385000  0.98875
4  Set4         20        0  0.000000   0.0000   0.5075 0.7450625  0.98375
5  Set5         20        0  0.000000   0.0000   0.5050 0.7266250  0.98375
6  Set6         20        0  0.000000   0.0000   0.5050 0.7928125  0.99625
7  Set7         20        0  0.000000   0.0000   0.5000 0.7905625  0.99625
8  Set8         20        0  0.000000   0.0000   0.5000 0.7828125  0.99375
9  Set9         20        0  0.000000   0.0000   0.6725 0.8363750  0.99375
     B.score     B.var     B.pval    S.pval     O.pval     O.minp O.minp.rho
1  194.05011  81468.56 0.49659373 0.1161530 0.18921934 0.11615304          0
2  -82.55532 275927.57 0.87511719 0.8984428 1.00000000 0.87511719          1
3  184.18466 236240.82 0.70472883 0.4849650 0.65744121 0.48496498          0
4  296.38608  26152.83 0.06684276 0.3678975 0.10658179 0.06684276          1
5  446.62340  74481.48 0.10173395 0.1360848 0.14697384 0.10173395          1
6  260.94739 129355.49 0.46812167 0.6745593 0.63135502 0.46812167          1
7  186.76451 144364.92 0.62304084 0.4675570 0.62998909 0.46755700          0
8 -217.12053 109511.48 0.51175878 0.0405403 0.07271573 0.04054030          0
9   32.51344 177820.67 0.93854153 0.3668321 0.55166531 0.36683209          0
      E.pval
1 0.18887302
2 0.96115053
3 0.50543496
4 0.11280646
5 0.30955819
6 0.57325088
7 0.56439534
8 0.05185726
9 0.59186820
```

It returns a data frame with first 8 columns showing the group (variant set) name, number of variants in each group, minimum, mean, maximum missing rate of variants in each group, minimum, mean, maximum effect allele frequency of variants in each group, followed by variant set test results. For Burden, 3 columns will be included to show the burden test score, variance of the score, and its P value. For SKAT, the P value column will be included. For SKAT-O, 3 columns will be included to show SKAT-O P value, minimum P value in the search grid, and the value of the mixing parameter ρ at which the minimum P value is observed. For the efficient hybrid test, the P value column will be included.

For a single study, intermediate files containing single variant scores and their covariance matrices for each variant set (based on the group definition file) can be saved for future use in re-analysis and/or meta-analysis. For example, here we perform the burden test and save intermediate files:

```
> glmm.rvtests(model0, group.file = group.file, geno.file = geno.file,
+         MAF.range = c(1e-7, 0.5), miss.cutoff = 1, method = "davies",
+         tests = "B", meta.file.prefix = "glmm.rvtests.meta")
```

```
# of selected samples: 400
  group n.variants miss.min miss.mean miss.max freq.min freq.mean freq.max
1  Set1         20        0  0.000875   0.0175   0.5000 0.8150402  0.99125
2  Set2         20        0  0.000000   0.0000   0.6400 0.8795625  0.99125
3  Set3         20        0  0.000000   0.0000   0.5675 0.8385000  0.98875
4  Set4         20        0  0.000000   0.0000   0.5075 0.7450625  0.98375
5  Set5         20        0  0.000000   0.0000   0.5050 0.7266250  0.98375
6  Set6         20        0  0.000000   0.0000   0.5050 0.7928125  0.99625
7  Set7         20        0  0.000000   0.0000   0.5000 0.7905625  0.99625
8  Set8         20        0  0.000000   0.0000   0.5000 0.7828125  0.99375
9  Set9         20        0  0.000000   0.0000   0.6725 0.8363750  0.99375
     B.score     B.var     B.pval
1  194.05011  81468.56 0.49659373
2  -82.55532 275927.57 0.87511719
3  184.18466 236240.82 0.70472883
4  296.38608  26152.83 0.06684276
5  446.62340  74481.48 0.10173395
6  260.94739 129355.49 0.46812167
7  186.76451 144364.92 0.62304084
8 -217.12053 109511.48 0.51175878
9   32.51344 177820.67 0.93854153
```

In the example above, a space-delimited file "glmm.rvtests.meta.score.1" will be generated to save the single variant scores, and a binary file "glmm.rvtests.meta.var.1" will be generated to save the covariance matrices for the variant sets. Note that the binary file is not human-readable, but can be used by **glmm.rvtests.meta** in re-analysis and/or meta-analysis.

5.3.2 Meta-analysis

With intermediate files generated by **glmm.rvtests**, the function **glmm.rvtests.meta** can be used in re-analysis of single study results, and/or meta-analysis to combine multiple studies. Here we show an example of rerunning SKAT using intermediate files generated above in the burden test:

```
> glmm.rvtests.meta(meta.files.prefix = "glmm.rvtests.meta", n.files = 1,
+         group.file = group.file, MAF.range = c(1e-7, 0.5),
+         miss.cutoff = 1, method = "davies", tests = "S")

  group n.variants    S.pval
1  Set1         20 0.1161530
2  Set2         20 0.8984428
3  Set3         20 0.4849650
4  Set4         20 0.3678975
5  Set5         20 0.1360848
6  Set6         20 0.6745593
7  Set7         20 0.4675570
8  Set8         20 0.0405403
9  Set9         20 0.3668321
```

The first argument, "meta.files.prefix", is a vector of intermediate files' prefix with length equal to the number of studies, and the second argument, "n.files", is a vector of integers showing how many sets of intermediate files each study has (also with length equal to the number of studies). In our above example of re-analysis, we have one set of intermediate files ("glmm.rvtests.meta.score.1" and "glmm.rvtests.meta.var.1") with prefix "glmm.rvtests.meta". The group definition file (passed to "group.file") should be the same as the one used to generate intermediate files by **glmm.rvtests** (with possibly different weights allowed). In the above example, only SKAT is performed, but four variant set tests are supported in the SMMAT framework: "B" for the burden test, "S" for SKAT, "O" for SKAT-O and "E" for the efficient hybrid test of the burden test and SKAT. You can include one or more tests by passing a vector to the argument "tests". If "O" is selected, the burden test and SKAT results will be automatically included; if "E" is selected, the burden test results will be automatically included.

6 Output

The single variant score test function **glmm.score** generates a tab-delimited plain text output file. Here we show the header and the first five rows of the example output "glmm.score.text.testoutfile.txt" from using a plain text genotype file "geno.txt" in the function **glmm.score**:

```
SNP     Allele1 Allele2 N       AF        SCORE    VAR     PVAL
SNP1    A       T       393     0.0254453 1.985    4.55635 0.352406
SNP2    A       C       400     0.5       3.51032  46.3328 0.60606
SNP3    A       C       400     0.2075    -0.5334  30.6023 0.923185
SNP4    A       G       400     0.29875   -3.11494 40.5128 0.624567
SNP5    A       G       400     0.59375   -4.00135 42.2757 0.538287
...
```

The first 3 columns are copied from the genotype file using "infile.ncol.print" with names specified using "infile.header.print". Results are included in 5 columns: the sample size N, the allele frequency (AF) of the effect allele (Allele2 in this example, but it is the user's choice: you can also define Allele1 as the effect allele in your coded genotype file), the score statistic SCORE of the effect allele, the variance of the score VAR, and score test P value PVAL.

If you use a PLINK binary PED file "geno.bed" as the genotype file, here are the header and the first 5 rows of the example output "glmm.score.bed.testoutfile.txt" from **glmm.score**:

```
CHR     SNP     cM      POS     A1      A2      N       AF
1       SNP1    0       1       T       A       393     0.974555
1       SNP2    0       2       A       C       400     0.5
1       SNP3    0       3       C       A       400     0.7925
1       SNP4    0       4       G       A       400     0.70125
1       SNP5    0       5       A       G       400     0.59375
...
SCORE    VAR     PVAL
-1.985   4.55635 0.352406
3.51032  46.3328 0.60606
```

```
0.5334   30.6023 0.923185
3.11494  40.5128 0.624567
-4.00135 42.2757 0.538287
...
```

The first 6 columns are copied from the .bim file (in our example, "geno.bim"): the chromosome CHR, SNP name, genetic location cM, physical position POS, and alleles A1 and A2. Results are included in 5 columns: the sample size N, the allele frequency (AF) of A2 allele, the score statistic SCORE of A2 allele, the variance of the score VAR, and score test P value PVAL.

If you use a GDS genotype file "geno.gds", here are the header and the first 5 rows of the example output "glmm.score.gds.testoutfile.txt" from **glmm.score**:

```
SNP     CHR     POS     REF     ALT     N       MISSRATE AF
SNP1    1       1       T       A       393     0.0175   0.974554707379135
SNP2    1       2       A       C       400     0        0.5
SNP3    1       3       C       A       400     0        0.7925
SNP4    1       4       G       A       400     0        0.70125
SNP5    1       5       A       G       400     0        0.59375
...
SCORE                   VAR                     PVAL
-1.98499783214996       4.55635423431794        0.352406178302761
3.51031635867184        46.3327709012496        0.606059815778752
0.533400489551472       30.6022850154632        0.923185358927845
3.11494101606552        40.5127615345412        0.62456656152105
-4.0013509071669        42.2757215856855        0.538287192500612
...
```

The first 5 columns are extracted from the GDS file: SNP ("annotation/id"), CHR ("chromosome"), POS ("position"), reference and alternate alleles ("allele"). Results are included in 6 columns: the sample size N (with non-missing genotypes), the genotype missing rate MISSRATE, the allele frequency (AF) of ALT allele, the score statistic SCORE of ALT allele, the variance of the score VAR, and score test P value PVAL.

The meta-analysis function **glmm.score.meta** generates a tab-delimited plain text output file. Here are the header and the first 5 rows of the example output from the meta-analysis "glmm.score.meta.testoutfile.txt":

```
SNP     A1      A2      N       AF      SCORE    VAR      PVAL
L14     A       C       10000   0.65895 21.5371  445.72   0.30766609304268
L25     A       C       10000   0.78425 14.3376  387.091  0.466163476535903
L7      A       C       20000   0.50435 33.1136  1019.122 0.299608382095623
L9      A       C       30000   0.39875 -33.0641 904.842  0.271687891048334
L35     A       C       10000   0.78425 14.3376  387.091  0.466163476535903
...
```

The first 3 columns are set by the function **glmm.score.meta** to denote SNP name and alleles (your choice of either A1 or A2 as the effect allele). N is the total sample size, AF is the effect allele frequency, SCORE is the summary score statistic of the effect allele, VAR is the variance of the summary score statistic, and PVAL is the meta-analysis P value.

In variant set tests **glmm.rvtests**, if "meta.file.prefix" is specified, space-delimited intermediate files for single variant scores and binary intermediate files for covariance matrices will be generated. Here are the header and the first 5 rows of the example intermediate file "glmm.rvtests.meta.score.1":

```
group chr pos ref alt N missrate altfreq
Set1 1 1 T A 393 0.0175 0.974554707379135
Set1 1 2 A C 400 0 0.5
Set1 1 3 C A 400 0 0.7925
Set1 1 4 G A 400 0 0.70125
Set1 1 5 A G 400 0 0.59375
...
SCORE VAR PVAL
-1.98499783213792 4.55635423431748 0.352406178305658
3.51031635867802 46.3327709012495 0.606059815778118
0.533400489561255 30.6022850154629 0.92318535892644
3.11494101607418 40.5127615345409 0.624566561520085
-4.00135090715957 42.2757215856853 0.538287192501356
...
```

The first 5 columns are copied from the group definition file, indicating the variant set (group) id, variant chromosome, variant position, variant reference allele, variant alternate allele, respectively. Results are included in 6 columns: the sample size N (with non-missing genotypes), the genotype missing rate missrate, the alt allele frequency altfreq, the score statistic SCORE of alt allele, the variance of the score VAR, and single variant score test P value PVAL.

7 Advanced options

7.1 Alternative model fitting algorithms

By default we use the Average Information REML algorithm[12,13] to fit the GLMM in **glmmkin**, which is computationally efficient and recommended in most cases. However, there are also alternative model fitting algorithms:

`method = "REML", method.optim = "Brent"`

It maximizes the restricted likelihood using the derivative-free Brent method,[14] but only works when there is one matrix for the covariance structure of the random effects.

`method = "ML", method.optim = "Brent"`

It maximizes the likelihood using the Brent method.

`method = "REML", method.optim = "Nelder-Mead"`

It maximizes the restricted likelihood using the Nelder-Mead method,[15] however it is usually very slow in large samples.

`method = "ML", method.optim = "Nelder-Mead"`

It maximizes the likelihood using the Nelder-Mead method.

Note that the default algorithm is

```
method = "REML", method.optim = "AI"
```

A maximum likelihood version of Average Information algorithm is not available in **glmmkin**.

7.2 Changing model fitting parameters

By default we set the maximum number of iteration to 500 and tolerance to declare convergence to 1e-5:

```
maxiter = 500, tol = 1e-5
```

These parameters can be changed. When using the Brent method for maximizing the likelihood (or restricted likelihood), we specify the search range of the ratio of the variance component parameter τ_k over the dispersion parameter ϕ to be between 1e-5 and 1e5, and we divide the search region evenly into 10 regions on the log scale:

```
taumin = 1e-5, taumax = 1e5, tauregion = 10
```

These parameters can also be changed, but they are only effective when using the Brent method.

7.3 Missing genotypes

It is recommended to perform genotype quality control prior to analysis to impute missing genotypes or filter out SNPs with high missing rates. However, *GMMAT* does allow missing genotypes, and imputes to the mean value by default. Alternatively, in **glmm.score** and **glmm.wald**, missing genotypes can be omitted from the analysis using

```
missing.method = "omit"
```

In variant set tests using **glmm.rvtests**, instead of imputing missing genotypes to the mean value, you can impute missing genotypes to 0 (homozygous reference allele) using

```
missing.method = "impute2zero"
```

If using a plain text (or compressed .gz and .bz2) genotype file, missing genotypes should be coded as "NA". If you have missing genotypes coded in a different way, you can specify this in the argument "infile.na".

7.4 Reordered genotypes

The genotype file (either a plain text file, a PLINK binary PED file, or a GDS file) can include more individuals than in the phenotype and covariates data frame or the matrices, and they can be in different orders. *GMMAT* handles this issue using an argument "select" in both **glmm.score** and **glmm.wald**. For example, if the order of individuals in your genotype file is A, B, C, D, but you only have 3 individuals in the phenotype and covariates data frame and the matrices (with order C, A, B), then you can specify

```
select = c(2, 3, 1, 0)
```

to reflect the order of individuals. Note that since individual D is not included in the phenotype and covariates data frame or the matrices, its order is assigned to 0. The length of the vector must match the number of individuals in your genotype file. The order of individuals in the phenotype and covariates data frame and the covariance structure matrices must be the same.

In variant set tests **glmm.rvtests**, the order of individuals is matched using the argument "null.obj.id" to indicate the ID of individuals in the **glmmkin** fitted object from the null GLMM, passed to the argument "null.obj". By default "null.obj.id" is NULL, **glmm.rvtests** will extract ID from "null.obj" using "id_include" returned in the **glmmkin** fitted null model object. The ID will be matched to "sample.id" in the GDS genotype file.

7.5 Parallel computing

Parallel computing can be enabled in **glmm.score** and **glmm.rvtests** using the argument "ncores" to specify how many cores you would like to use on a computing node. By default "ncores" is 1, meaning that these functions will run in a single thread. Currently parallel computing is only implemented for GDS format genotype files.

If you enable parallel computing and save intermediate files, you will get multiple sets of intermediate files. For example, if your "ncores" is 12 and you specified "meta.file.prefix" to "study1", then you will get 12 sets of (totaling 24) intermediate files "study1.score.1", "study1.var.1", "study1.score.2", "study1.var.2", ..., "study1.score.12", "study1.var.12". Later in the meta-analysis to combine with 2 sets of intermediate files "study2.score.1", "study2.var.1", "study2.score.2", "study2.var.2", you will need to use

```
meta.files.prefix = c("study1", "study2"), n.files = c(12, 2)
```

If your R is configured with Intel MKL and you would like to enable parallel computing, it is recommended that you set the environmental variable "MKL_NUM_THREADS" to 1 before running R to avoid hanging. Alternatively, you can do this at the beginning of your R script by using

```
> Sys.setenv(MKL_NUM_THREADS = 1)
```

7.6 Variant filters

Variants can be filtered in **glmm.score** and **glmm.rvtests** based on minor allele frequency (MAF) and missing rate filters. The argument "MAF.range" specifies the minimum and maximum MAFs for a variant to be included in the analysis. By default the minimum MAF is 1×10^{-7} and the maximum MAF is 0.5, meaning that only monomorphic markers in the sample will be excluded (if your sample size is no more than 5 million). The argument "miss.cutoff" specifies the maximum missing rate for a variant to be included in the analysis. By default it is set to 1, meaning that no variants will be removed due to high genotype missing rates.

7.7 Internal minor allele frequency weights

Internal weights are calculated based on the minor allele frequency (NOT the effect allele frequency, therefore, variants with effect allele frequencies 0.01 and 0.99 have the same weights) as a beta probability density function. Internal weights are multiplied by the external weights given in the last column of the group definition file. To turn off internal weights, use

```
MAF.weights.beta = c(1, 1)
```

to assign flat weights, as a beta distribution with parameters 1 and 1 is a uniform distribution on the interval between 0 and 1.

7.8 Allele flipping

In variant set tests **glmm.rvtests**, by default the alt allele is used as the coding allele and variants in each variant set are matched strictly on chromosome, position, reference and alternate alleles.

The argument "auto.flip" allows automatic allele flipping if a specified variant is not found in the genotype file, but a variant at the same chromosome and position with reference allele matching the alternate allele in the group definition file "group.file", and alternate allele matching the reference allele in the group definition file "group.file", to be included in the analysis. Please use with caution for whole genome sequence data, as both ref/alt and alt/ref variants at the same position are not uncommon, and they are likely two different variants, rather than allele flipping.

The argument "use.minor.allele" allows using the minor allele instead of the alt allele as the coding allele in variant set tests. Note that this choice does not change "S" for SKAT results, but "B" for the burden test, "O" for SKAT-O and "E" for efficient hybrid test of the burden test and SKAT results will be affected. Generally the alt allele can either be the minor or the major allele. If in a variant set, different variants with alt allele frequencies 0.001 and 0.998 are combined together in a burden test, the results would be difficult to interpret. We generally recommend turning on the "use.minor.allele" option, unless you know the ancestry alleles explicitly and the specific scientific hypothesis clearly that you would like to test. Along with the MAF filter, this option is useful for combining rare mutations, assuming rare allele effects are in the same direction.

7.9 *P* values of weighted sum of chi-squares

In variant set tests **glmm.rvtests**, you can use 3 methods in the "method" argument to compute P values of weighted sum of chi-square distributions: "davies",[16] "kuonen"[17] and "liu".[18] By default "davies" is used, if it returns an error message in the calculation, or a P value greater than 1, or less than 1×10^{-5}, "kuonen" method will be used. If "kuonen" method fails to compute the P value, "liu" method will be used.

7.10 Heterogeneous genetic effects in variant set meta-analysis

Heterogeneous genetic effects[19] are allowed in variant set tests meta-analysis function **glmm.rvtests.meta**, by specifying groups using the "cohort.group.idx" argument. By default all studies are assumed to share the same genetic effects in the meta-analysis, and this can be changed by assigning different group indices to studies. For example,

```
cohort.group.idx = c("a","b","a","a","b")
```

means cohorts 1, 3, 4 are assumed to have homogeneous genetic effects, and cohorts 2, 5 are in another group with homogeneous genetic effects (but possibly heterogeneous with group "a").

7.11 Other options

By default, genotypes are centered to the mean before the analysis in single variant tests. You can turn this feature off by specifying

```
center = FALSE
```

in both **glmm.score** and **glmm.wald** functions to use raw genotypes.

If your genotype file is a plain text (or a compressed .gz and .bz2 file), and you want to read in fewer lines than all lines included in the file, you can use the "infile.nrow" argument to specify how many lines (including lines to be skipped using "infile.nrow.skip") you want to read in. By default the delimiter is assumed to be a tab, but you can change it using the "infile.sep" argument. These options are implemented in **glmm.score** and **glmm.wald**.

In the score test function **glmm.score**, by default 100 SNPs are tested in a batch. You can change it using the "nperbatch" argument, but the computational time can increase substantially if it is either too small or too large, depending on the performance of your system.

If you perform Wald tests **glmm.wald** and use a plain text (or a compressed .gz and .bz2) file, and your SNPs are not in your first column, you can change "snp.col" in **glmm.wald** to indicate which column is your SNP name.

In the variant set tests **glmm.rvtests**, by default the group definition file "group.file" should be tab delimited, but you can change it using the "group.file.sep" argument. Also there is a "Garbage.Collection" argument (default FALSE), if turned on, **glmm.rvtests** will call the function **gc** for each variant set tested. It helps save memory footprint, but the computation speed might be slower.

8 Version

8.1 Version 0.6 (October 12, 2015)

Initial public release of *GMMAT*.

8.2 Version 0.7 (January 22, 2016)

1. Merged old functions **glmm.score.text** and **glmm.score.bed** to **glmm.score**.
2. Merged old functions **glmm.wald.text** and **glmm.wald.bed** to **glmm.wald**.
3. **glmm.score** now takes "obj", a glmmkin class object returned from **glmmkin** to set up the score test, instead of "res" and "P" from old functions **glmm.score.text** and **glmm.score.bed**.

4. Implemented model fitting with fixed variance components in **glmmkin**, and model refitting when variance component estimates are on the boundary of the parameter space, and default unconverged Average Information REML to derivative-free Brent method (one variance component parameter) or Nelder-Mead method (more than one variance component parameters).

5. Implemented offset in **glmmkin**.

6. Renamed alpha, eta, mu to coefficients, linear.predictors, fitted.values in **glmmkin** returned object.

7. Implemented score test meta-analysis function **glmm.score.meta**.

8. Fixed minor bugs in **glmm.wald** to handle errors in fitting each alternative GLMM, and minor bugs in fitting alternative GLMMs using derivative-free algorithms.

8.3 Version 0.7-1 (January 22, 2016)

Light version of v0.7: same features as v0.7 except that this light version does not depend on the C++ library boost and cannot take compressed plain text files .gz and .bz2 as genotype files.

8.4 Version 0.9 (March 9, 2018)

1. Added "id_include" to **glmmkin** returned object to indicate which rows in the data have nonmissing outcome/covariates and are included in the model fit, which is useful to create

   ```
   > select <- match(geno_ID, pheno_ID[obj$id_include])
   > select[is.na(select)] <- 0
   ```

   that can be used as the "select" argument in **glmm.score**.

2. Removed memory duplicates for big matrices in C++ and R code.

3. Support for GDS format genotype files implemented in functions **glmm.score** and **glmm.wald**, with optional parallel computing.

4. MAF.range and miss.cutoff implemented in **glmm.score** for minor allele frequency and missing rate filters.

5. Implemented variant set tests **glmm.rvtests**, including burden test, SKAT, SKAT-O and SMMAT, with optional parallel computing.

6. Implemented variant set re-analysis/meta-analysis function **glmm.rvtests.meta**.

7. Implemented heteroscedastic linear mixed models in **glmmkin** and **glmm.wald** by specifying "groups".

8.5 Version 0.9.1 (May 13, 2018)

1. In variant set tests **glmm.rvtests** and **glmm.rvtests.meta**, tests "Burden", "SKAT", "SKAT-O" and "SMMAT" changed to "B", "S", "O" and "E", respectively, to denote 4 variant set tests in the SMMAT framework: the burden test, SKAT, SKAT-O and the efficient hybrid test of the burden test and SKAT.

2. Implemented known weights (e.g. binomial denominator) in **glmmkin** and **glmm.wald** by passing the argument "weights" to the **glm** object "fixed"

3. Implemented internal MAF-based weights, using the argument "MAF.weights.beta" to denote the two beta probability density function parameters. Note that the weights are calculated on the minor allele frequency (not the effect allele frequency). Internal weights are multiplied by the external weights given in the last column of the group definition file.

8.6 Version 0.9.2 (June 8, 2018)

1. In variant set tests **glmm.rvtests** and **glmm.rvtests.meta**, test "O" (SKAT-O in the SMMAT framework) p-value switched to minimum p-value multipled by the number of points on the search grid if the integration result is larger than the latter (consistent with implementation in the SKAT package).

9 Contact

Please refer to the R help document of *GMMAT* for specific questions about each function. For comments, suggestions, bug reports and questions, please contact Han Chen (Han.Chen.2@uth.tmc.edu). For bug reports, please include an example to reproduce the problem without having to access your confidential data.

10 Acknowledgments

We would like to thank Dr. Chaolong Wang and Dr. Brian Cade for comments and suggestions on *GMMAT* and the user manual. We would also like to thank Dr. Matthew Conomos for help with the Average Information REML algorithm, Dr. Stephanie Gogarten for help with the GDS genotype format, and Jennifer Brody for help with parallel computing and App development in Analysis Commons, a cloud computing platform. The *GMMAT* implementation is supported by NIH grant R00 HL130593, and the analysis pipeline implementation (the gmmat App) in Analysis Commons is supported by NIH grant U01 HL120393.

References

[1] Breslow, N. E. and Clayton, D. G. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association* **88,** 9–25 (1993).

[2] Morgenthaler, S. and Thilly, W. G. A strategy to discover genes that carry multi-allelic or mono-allelic risk for common diseases: A cohort allelic sums test (CAST). *Mutation Research* **615,** 28–56 (2007).

[3] Li, B. and Leal, S. M. Methods for detecting associations with rare variants for common diseases: Application to analysis of sequence data. *The American Journal of Human Genetics* **83,** 311–321 (2008).

[4] Madsen, B. E. and Browning, S. R. A groupwise association test for rare mutations using a weighted sum statistic. *PLOS Genetics* **5,** e1000384 (2009).

[5] Morris, A. P. and Zeggini, E. An evaluation of statistical approaches to rare variant analysis in genetic association studies. *Genetic Epidemiology* **34,** 188–193 (2010).

[6] Wu, M. C., Lee, S., Cai, T., Li, Y., Boehnke, M. and Lin, X. Rare-variant association testing for sequencing data with the sequence kernel association test. *The American Journal of Human Genetics* **89,** 82–93 (2011).

[7] Lee, S., Wu, M. C. and Lin, X. Optimal tests for rare variant effects in sequencing association studies. *Biostatistics* **13,** 762–775 (2012).

[8] Chen, H., Wang, C., Conomos, M. P., Stilp, A. M., Li, Z., Sofer, T., Szpiro, A. A., Chen, W., Brehm, J. M., Celedón, J. C., Redline, S., Papanicolaou, G. J., Thornton, T. A., Laurie, C. C., Rice, K. and Lin, X. Control for Population Structure and Relatedness for Binary Traits in Genetic Association Studies via Logistic Mixed Models. *The American Journal of Human Genetics* **98,** 653–666 (2016).

[9] Brody, J. A., Morrison, A. C., Bis, J. C., O'Connell, J. R., Brown, M. R., Huffman, J. E., Ames, D. C., Carroll, A., Conomos, M. P., Gabriel, S., Gibbs, R. A., Gogarten, S. M., Gupta, N., Jaquish, C. E., Johnson, A. D., Lewis, J. P., Liu, X., Manning, A. K., Papanicolaou, G. J., Pitsillides, A. N., Rice, K. M., Salerno, W., Sitlani, C. M., Smith, N. L., NHLBI Trans-Omics for Precision Medicine (TOPMed) Consortium, The Cohorts for Heart and Aging Research in Genomic Epidemiology (CHARGE) Consortium, TOPMed Hematology and Hemostasis Working Group, CHARGE Analysis and Bioinformatics Working Group, Heckbert, S. R., Laurie, C. C., Mitchell, B. D., Vasan, R. S., Rich, S. S., Rotter, J. I., Wilson, J. G., Boerwinkle, E., Psaty, B. M. and Cupples, L. A. Analysis commons, a team approach to discovery in a big-data environment for genetic epidemiology. *Nature Genetics* **49,** 1560–1563 (2017).

[10] Zhou, X. and Stephens, M. Genome-wide efficient mixed-model analysis for association studies. *Nature Genetics* **44,** 821–824 (2012).

[11] Conomos, M. P., Laurie, C. A., Stilp, A. M., Gogarten, S. M., McHugh, C. P., Nelson, S. C., Sofer, T., Fernández-Rhodes, L., Justice, A. E., Graff, M., Young, K. L., Seyerle, A. A., Avery, C. L., Taylor, K. D., Rotter, J. I., Talavera, G. A., Daviglus, M. L., Wassertheil-Smoller, S., Schneiderman, N., Heiss, G., Kaplan, R. C., Franceschini, N., Reiner, A. P., Shaffer, J. R., Barr, R. G., Kerr, K. F., Browning, S. R., Browning, B. L., Weir, B. S., Avilés-Santa, M. L., Papanicolaou, G. J., Lumley, T., Szpiro, A. A., North, K. E., Rice, K., Thornton, T. A. and Laurie, C. C. Genetic Diversity and Association Studies in US Hispanic/Latino Populations: Applications in the Hispanic Community Health Study/Study of Latinos. *The American Journal of Human Genetics* **98,** 165–184 (2016).

[12] Gilmour, A. R., Thompson, R. and Cullis, B. R. Average information REML: an efficient algorithm for variance parameter estimation in linear mixed models. *Biometrics* **51,** 1440–1450 (1995).

[13] Yang, J., Lee, S. H., Goddard, M. E. and Visscher, P. M. GCTA: a tool for genome-wide complex trait analysis. *The American Journal of Human Genetics* **88,** 76–82 (2011).

[14] Brent, R. P. Chapter 4: An Algorithm with Guaranteed Convergence for Finding a Zero of a Function, Algorithms for Minimization without Derivatives, Englewood Cliffs, NJ: Prentice-Hall, ISBN 0-13-022335-2 (1973).

[15] Nelder, J. A. and Mead, R. A simplex algorithm for function minimization. *Computer Journal* **7,** 308–313 (1965).

[16] Davies, R. B. Algorithm AS 155: The Distribution of a Linear Combination of χ^2 Random Variables. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* **29,** 323–333 (1980).

[17] Kuonen, D. Saddlepoint Approximations for Distributions of Quadratic Forms in Normal Variables. *Biometrika* **86,** 929–935 (1999).

[18] Liu, H., Tang, Y. and Zhang, H. H. A new chi-square approximation to the distribution of non-negative definite quadratic forms in non-central normal variables. *Computational Statistics & Data Analysis* **53,** 853–856 (2009).

[19] Lee, S., Teslovich, T. M., Boehnke, M. and Lin, X. General Framework for Meta-analysis of Rare Variants in Sequencing Association Studies. *The American Journal of Human Genetics* **93,** 42–53 (2013).