## Overview

This package contains the MATLAB algorithms used in the paper **Heritability and Genetic Correlations Explained by Common SNPs for Metabolic Syndrome Traits** (see citation below). The main purpose of this paper was to estimate the narrow-sense heritability ($h^2$) for biological traits (phenotypes) and the genetic and residual correlations ($r_{g(e)}$) between them for any number of traits. $h^2$ is a common measure that accounts for the fraction of inheritance between parents and offspring due to additive genetic variants. In addition to these measures and by taking advantage of the relationship between correlation length and pedigree, we estimated the genomic contribution due to common genetic variants (prevalence of 5% or more in the population) compared to all genetic variants. *The scripts are written for genomic analysis but are not specific to this type of data and be easily adapted to other data sources. Other measurement matrices and quantitative outcome variables may be used; though, keep in mind we assume the columns of both are standard normalized.*

The major solver is the restricted maximum Likelihood expectation maximization scheme (REML EM). These algorithms are designed for the simultaneous estimate of variances and covariance among any number of traits and is only limited by computational resources. We emphasize that it is capable of going well beyond a bivariate analysis using realistic resources as we illustrated in the paper by the simultaneous analysis of seven Metabolic Syndrome traits. Such simultaneous analysis takes advantage of additional information to improve estimates, even for the single variance components used to estimate $h^2$. The major implementations are done using four functions: 1) $REMLEM$, 2) $FisherInv$, 3) *calch*2, and 4) *calccorr*. $REMLEM$ is used to estimate the mean genetic and residual variances, and cross-trait covariances. $FisherInv$ calculates the inverse Fisher information needed for error estimates.

The script *mvlmesim* shows an example of a full implementation. For this we include simulated data with correlated outcome variables. The $genZA$ script also shows how the genotype matrix is normalized using the vector of minor allele frequencies as written in the paper.

*Note there may be some discrepancy between the variable notation here and what is written within the actual functions. The notation used here is more consistent with the general use of these variables. If you want to follow the variables within the functions then match the variable names by the argument order. In future releases the names will be consistent.*

## Credits

The functions $REMLEM$, $REMLEMtrans$, $FisherInv$, and *calccorr* were originally written by Carson C Chow and edited by Shashaank Vattikuti. Both authors are from the Laboratory of Biological Modeling at the National Institutes of Health, NIDDK.

## Citation

If you use these scripts for any published work, please cite the manuscript describing the methods. The paper is publicly available at: `http://www.ncbi.nlm.nih.gov/pubmed/22479213`.

Vattikuti S, Guo J, Chow CC. Heritability and genetic correlations explained by common SNPs for metabolic syndrome traits. PLoS Genet. 2012;8(3):e1002637.

## Tips

Here are few suggestions for getting the most from this analysis. We have observed two major issues with using REML EM straightoff: a dependence on initial conditions and the potential to oscillate between solutions. Both are common problems in machine learning. Regarding initial conditions, the major issue with the full EM algorithm is an inability for the covariances to cross zero; therefore, if they are initialized with the wrong sign then the fits will converge near zero but not cross it, and thus resulting in a poor fit. This is shown in Figure 1. The actual genetic correlation is minus 0.5 and that of for the residuals is positive 0.3. When they are initialized with the incorrect sign (left plots), the fits can get trapped on the wrong side of zero. The top-left is when the residual covariance ($\hat{r}_e$) is initialized with the wrong sign, and the bottom-left shows when both are incorrectly initialized. The plots on the right show the fits when initialized with the correct signs (final iteration for both: $\hat{r}_g$=-0.6 (se=0.1), $\hat{r}_e$=0.31 (se=0.08)).

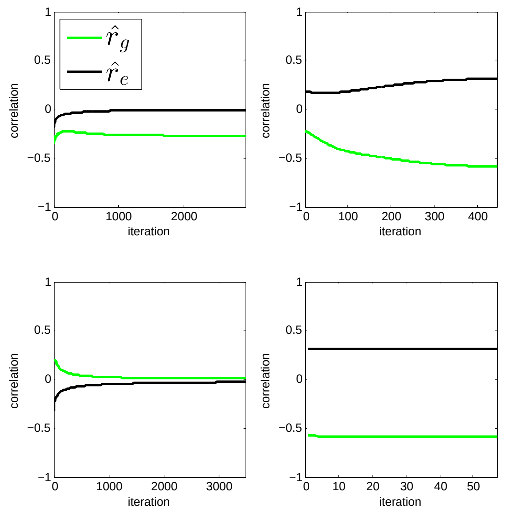

**Figure 1**

We suggest two solutions:

1. If the final fit is near zero or equivalently if you plot the fits over iterations and the path seems to be converging on zero, then run $REMLEM$ again but change the sign and see if you get a better fit.

2. A variant of the REML EM method that uses a diagonal transformation of the covariance matrix can cross zero (implemented in $REMLEMtrans$). As such, use this to find initial covariances (particularly the signs) and then follow up with $REMLEM$ to get the final fits. We advise always following up/finishing with the non-transform version $REMLEM$.

We evaluated a particular pathological case and solution (2) works the majority of the time. Rarely $REMLEMtrans$ will provide poor initial conditions; though, the probability of this decreases with greater power. In Figure 1 the poor fits on the left are taken from rare instances when $REMLEMtrans$ fails to get the correct signs. The bottom-right plot is the more typical case where not only are the signs correct but the initial conditions are very close to the eventual $REMLEM$ fits ($REMLEMtrans$ was not used for the top-right). Given the occassional pathologic initial conditions, it may be useful to always check whether $REMLEM$ is stuck and flip the sign as discussed in (1) above. We also noticed that the number of iterations may be oddly large for such cases versus good cases. This may be an additional indication of this problem.

Here is one approach for a full implementation of a multivariate model. First estimate the single trait variances using the univariate REML EM model. Then estimate the covariances using the $REMLtrans$ bivariate model followed by $REMLEM$. If the goal is to fit a mutlivariate model with more than two traits, then after accounting for the zero-pinning issue use the bivariate estimates for initial conditions on the full REML model. The script *mvlmesim* gives an example of the full implementation for a bivariate model.

Regarding oscillations, we have included a scaling parameter that adjusts the step size for $REMLEM$. The default is one and decreasing this may help.

Final short note, *dl* may not be monotonic as the multivariate REML algorithms fit the different parameter dimensions.

## Data Inputs

Here is a brief description of the data and preprocessing needed for $REMLEM$. (For non-genomic applications make the appropriate mental substitutions.) For details see Methods in the paper.

## The genomic correlation matrix, A.

The matrix **A** is generated by three steps, with the major step of calculating the covariance (correlation) matrix **A** in people space from a normalized measurement matrix. For the paper we restricted the analysis to binary measurements (genetic variants) (though this may be relaxed as long as the measruement columns are normalized correctly):

1. Generate the genotype matrix **G** by counting the number of minor alleles for each individual for each genetic measure. This will results in an $m \times n$ matrix, where $m$ are the number of subject rows and $n$ the number of measurement (e.g., SNP) columns. For the non-geneticist readers, a geneticist often starts with a set of genetic variants called *alleles* for each measurement on an individual. These may be considered two draws from a bi(multi)nomial distribution. In the additive model, we assume that summing across the set is linear and gives the overall magnitude (dose) along this measure. **G** then is the condensed measurement matrix based on the allele calls. Non-additive effects are thought to acrue in the residual contribution to the outcome variable variance.

2. Normalize **G** to generate **Z** (starting point for non-genomic applications). This is the standard operation of normalizing the measurement columns to have zero mean and unit variance. This may be done by calculating the empirical mean for each column and subtracting this from the appropriate column elements and similarily dividing them by the empirical standard deviation. Alternatively, you can estimate the theoretical mean and deviation using Binomial theory and the minor allele frequency (see Methods of the paper). We perform the normalization in the absence of missing genotypes and then replace missing values with the zero mean.

3. Calculate the genomic correlation matrix, **A**. This is done by taking the inner product of $\mathbf{ZZ}'$ and normalizing by the number of measures ($\mathbf{A}=\frac{\mathbf{ZZ}'}{n}$). (The diagonal may be adjusted for inbreeding as discussed in the Methods.) The resultant matrix **A** is an $m \times m$ matrix. For an example of how this is done, see the script for *genZA* used in *mvlmesim*.

## The covariate (fixed effects) matrix, X.

**X** accounts for covariates (fixed effects) that are projected out in REML. It is an $m \times n$ matrix (different than above), with m=$\text{N}_{\text{subjects}}$ multiplied by $\text{N}_{\text{traits}}$ and n=$\text{N}_{\text{fixed effects}}$ multiplied by $\text{N}_{\text{traits}}$ (N=number of). For example, in the case of $\text{N}_{\text{fixed effects}}$=1 (where there are no covariates) and $\text{N}_{\text{subjects}}$=2 and $\text{N}_{\text{traits}}$=3 the X matrix is:

$$X=\begin{pmatrix}1 & 0 & 0\\ 1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0 & 1\end{pmatrix}$$

If there is an additional fixed effect, such as age, then the matrix in the case of $\mathrm{N}_{\text{fixed effects}}$=2 and $\mathrm{N}_{\text{subjects}}$=2 and $\mathrm{N}_{\text{traits}}$=3 would look like this:

$$X=\begin{pmatrix}1 & age_{subj1} & 0 & 0 & 0 & 0\\ 1 & age_{subj2} & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & age_{subj1} & 0 & 0\\ 0 & 0 & 1 & age_{subj2} & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & age_{subj1}\\ 0 & 0 & 0 & 0 & 1 & age_{subj2}\end{pmatrix}$$

Note at the minimum you can run **X** as in the first case with no covariates if these have been removed prior to the analysis; though, this reduces the benefits of using a restricted maximum Likelihood approach. We use a function *genX* to generate the **X** matrix by feeding it the number of traits and the submatrix of covariates. In the example above, this is the submatrix formed by rows and columns 1 and 2. *genX* then tiles this submatrix along the diagonal given the number of traits.

## The phenotype (trait) matrix, Y.

The phenotype matrix **Y** has dimensions $m \times n$, where $m$=$\mathrm{N}_{\text{subjects}}$ and $n$=$\mathrm{N}_{\text{traits}}$. Each column consists of the phenotype measurements over all subjects for one trait. We standard normalize the columns of **Y** to have zero mean and unit variance. **It is important to record the column index order for the traits fed to *REMLEM*.** This will be needed to interpret the results.

## *REMLEM*

*REMLEM* runs a single iteration of the REML EM algorithm taking the genetic and residual covariance matrices at step i and returning the updated estimates at i+1. The function is intended to be cycled (iterated) until the tolerance condition (*tol*) is met (e.g., change in log-likelihood (*dl*) $< 1 \cdot 10^{-4}$). An example of the implementation looks like this:

```
dl=1;
```

```
tol=1*10^-4;

while dl>tol

[dl,Sigma_g,Sigma_e]=REMLEM(A,Y,X,Sigma_g,Sigma_e,(a));

end
```

We arbitrarily initialize dl to be one for the first iteration of the *while* loop; needs to be some number greater than *tol*. The arguments $\mathbf{A}, \mathbf{Y},$ and $\mathbf{X}$ are described in the section entitled *Data Inputs*. The argument $a$ is optional and adjusts the learning rate. The default is one, but if *dl* oscillates instead of converging then you can try lowering it. Sigma_g and Sigma_e are the major matrices that we are trying to determine. They are square matrices of dimensions $m \times m$, where $m$=$\mathrm{N}_{\mathrm{traits}}$. The diagonals are the variance estimates for the traits in column order of $\mathbf{Y}$. The off-diagonals are the respective trait covariances. As mentioned in the section *Tips*, how these are initialized can have a major affect. Below is one scheme we use in *mvlmesim*, which is later updated using variances from the univariate models and covariances from $REMLEMtrans$:

```
%random initial covariance matrices

offdiag=(1-eye(Ntr)).*(rand(Ntr)-.5); %initialize random covariances between -.5 and .5

offdiag=triu(offdiag)+triu(offdiag,1)'; %symmetrize matrix

Sigma_g=repmat(rand([Ntr,1]),[1 Ntr]).*eye(Ntr); %random initial variances

Sigma_e=(1-Sigma_g).*eye(Ntr); %inital sigma e, assuming pheno variance is one

Sigma_g=Sigma_g +offdiag; % compile random diagonals with random off

offdiag=(1-eye(Ntr)).*(rand(Ntr)-.5); %get random off-diagonals for sigma e

offdiag=triu(offdiag)+triu(offdiag,1)';

Sigma_e=Sigma_e +offdiag;
```

## $REMLEMtrans$

This is implemented as for $REMLEM$ except there is no optional learning rate parameter.

## *FisherInv*

*FisherInv* returns the inverse of the Fisher information matrix (**Finv**). This is necessary for estimating the error of the parameter estimates from *REMLEM* and for propagating the error in the $h^2$ and $r_{g(e)}$ calculations. An example of the implementation is this:

[Finv,Finvmap]=FisherInv(A,X,Sigma_g,Sigma_e)

**Finv** is a rather complicated output matrix. **Finv** consists of calculations on all combinations of the parameters estimated in *REMLEM*. For example in the simplest case of a single trait analysis, we are calculating **Finv** over combinations of two parameters: var(g) and var(e). We will call these $\theta_g$ and $\theta_e$ for clarity. The resultant matrix is a two-by-two matrix consisting of calculations on these parameter pairs:

$$\mathbf{Finv} \sim \begin{pmatrix} \theta_g,\theta_g & \theta_g,\theta_e \\ \theta_e,\theta_g & \theta_e,\theta_e \end{pmatrix}$$

Note the square-root of the element labeled $\theta_g,\theta_g$ is the estimated standard error of $\theta_g$ (var(g)). Similarily, the square-root of element $\theta_e,\theta_e$ is the estimated standard error of var(e). The off-diagonal element is the shared error between the parameters var(g) and var(e). For $h^2$ calculation this last element along with the others are used to propogate the error using a first-order Taylor expansion.

For two traits we estimated six parameters using *REMLEM*: var(g1), var(g2), cov(g1,g2), var(e1), var(e2), and cov(e1,e2). Again for clarity we will call these $\theta_{g1}$, $\theta_{g2}$, $\theta_{g1g2}$, $\theta_{e1}$, $\theta_{e2}$, and $\theta_{e1e2}$. This results in a 6-by-6 inverse Fisher information matrix with elements:

$$\mathbf{Finv} \sim \begin{pmatrix} \theta_{g1},\theta_{g1} & \theta_{g1},\theta_{e1} & \theta_{g1},\theta_{g1g2} & \theta_{g1},\theta_{e1e2} & \theta_{g1},\theta_{g2} & \theta_{g1},\theta_{e2} \\ \theta_{e1},\theta_{g1} & \theta_{e1},\theta_{e1} & \theta_{e1},\theta_{g1g2} & \theta_{e1},\theta_{e1e2} & \theta_{e1},\theta_{g2} & \theta_{e1},\theta_{e2} \\ \theta_{g1g2},\theta_{g1} & \theta_{g1g2},\theta_{e1} & \theta_{g1g2},\theta_{g1g2} & \theta_{g1g2},\theta_{e1e2} & \theta_{g1g2},\theta_{g2} & \theta_{g1g2},\theta_{e2} \\ \theta_{e1e2},\theta_{g1} & \theta_{e1e2},\theta_{e1} & \theta_{e1e2},\theta_{g1g2} & \theta_{e1e2},\theta_{e1e2} & \theta_{e1e2},\theta_{g2} & \theta_{e1e2},\theta_{e2} \\ \theta_{g2},\theta_{g1} & \theta_{g2},\theta_{e1} & \theta_{g2},\theta_{g1g2} & \theta_{g2},\theta_{e1e2} & \theta_{g2},\theta_{g2} & \theta_{g2},\theta_{e2} \\ \theta_{e2},\theta_{g1} & \theta_{e2},\theta_{e1} & \theta_{e2},\theta_{g1g2} & \theta_{e2},\theta_{e1e2} & \theta_{e2},\theta_{g2} & \theta_{e2},\theta_{e2} \end{pmatrix}$$

The return parameter **Finvmap** provides similar labels for the elements in **Finv**. As noted above, many of these elements are used for subsequent calculations such as $h^2$; however, you may be interested in reporting the errors on the *REMLEM* parameters themselves. For example, you may want to report the error on var(g), var(e), and cov(g1,g2). **Finvmap** can be used to find these elements and is especially useful when analyzing results from a large (many traits) multivariate REML EM run. These are labels for important elements that you may want to know, report:

gigi_gigi=var(var(gi))

eiei_eiei=var(var(ei))

gigj_gigj=var(cov(gi,gj))

eiej_eiej=var(cov(ei,ej))

Above, $i$ is the numeric label for the first trait and $j$ for the second trait (assign the smaller label to $i$). For example if you ran a simultaneous analysis of seven traits and want to know the variance on the estimate of the genetic covariance between traits 1 and 6, then you will want the element labeled g1g6_g1g6. Here is an example of how you can find the index for an element of interest (e.g., var(cov(gi,gj))) from **Finvmap** and extract its value from **Finv**:

```
tri=1;

trj=6;

lab=['g' num2str(tri) 'g' num2str(trj) '_' 'g' num2str(tri) 'g'];

num2str(trj)];

ind=find(strcmp(Finvmap, lab)==1);

var_covgigj=Finv(ind);

se_covgigj=sqrt(var_covgigj);
```

Note that *calch*2 and *calccorr* have similar extraction lines built-in and so you just need to feed them **Finv** and **Finvmap**.

## *calch*2

Having estimated the parameters using REMLEM and the Fisher information, you can now estimate $h^2$ and its error. Here is an example of how to implement the function:

```
[h2,h2se]=calch2(tr,Sigma_g,Sigma_e,Finv,Finvmap);
```

## *calccorr*

The genetic or residual correlations between traits is calculated using *calccorr*. The implementation is such:

```
[r,rse,rpval]=calccorr(tri,trj,eclass,Sigma,Finv,Finvmap)
```

As noted above, $i$ and $j$ are the numeric labels for the traits of interest (assign the smaller label to $i$). $eclass$ is either 'g' or 'e' and stands for the effect class you are interested in calculating; genetic or residual correlation. $Sigma$ is the appropriate covariance matrix, either genetic or residual, and **Finv** and **Finvmap** are the full matrices (has all genetic and residual elements) calculated above.