MTM Package

Contents

Introduction

MTM (Multiple-Trait Model) fits Bayesian Multivariate Gaussian models with an arbitrary number of random effects using a Gibbs sampler. The data equation for the jth trait (j = 1, ...) is expressed as

$$y_j = 1\mu_j + X\beta_j + u_{j1} + \ldots + u_{jq} + \varepsilon_j$$

where: $y_j = (y_{1j}, \ldots, y_{nj})'$ is an n-dimensional vector of phenotypes (`NA`s are allowed) with y_{ij} representing the phenotypic record of the ith subject for the jth trait, μ_j is an intercept, X a design matrix for fixed effects (the same effects are assumed for all traits) and β_j the corresponding vector of effects for the jth trait, $u_{jk} = (u_{1jk}, \ldots, u_{njk})', (k = 1, \ldots, q)$ is an n-dimensional vector of random effects, and $\varepsilon_j = (\varepsilon_{1j}, \ldots, \varepsilon_{nj})'$ is a vector of model residuals.

Conditional distribution of the data

Model residuals are assumed to follow a multivariate normal distribution, with null mean and covariance matrix $\text{Cov}(\varepsilon_1', \ldots, \varepsilon_n')' = R_0 \otimes I$ where R_0 is an $r \times r$ (within-subject) covariance matrix of model residuals and I is an n-dimensional identity matrix. Therefore, the conditional distribution of the data given the location and dispersion parameters is

$$p(y_1, \ldots, y_r | \beta_1, \ldots, u_1, \ldots, u_r) = \prod_{i=1}^{n} \text{MVN}(y_i | \eta_i, R_0)$$

where MVN(.|.,.) denotes a multivariate-normal density with mean η_i and covariance matrix R_0; here, η_i is an r-dimensional vector whose entries are the expected phenotypic values of the ith individual for each of the traits, that is

$$\eta_i = \begin{bmatrix} \eta_{1i} \\ \vdots \\ \eta_{ri} \end{bmatrix} = \begin{bmatrix} \mu_1 + x_1'\beta_1 + u_{i11} + \ldots + u_{i1q} \\ \vdots \\ \mu_r + x_1'\beta_r + u_{ir1} + \ldots + u_{irq} \end{bmatrix}$$

Prior distribution

The prior distribution is structured hierarchically. The first level of the prior specifies the distribution of the fixed and random effects given the codispersion parameters (the covariance matrices of the random effects, see below). The priors for the codispersion parameters are specified in a deeper level of the hierarchy.

The **intercepts and vectors of fixed effects** are assigned flat prior (each unknown is assigned a Gaussian prior with null mean and very large variance).

The vectors of **random effects** are assigned independent multivariate normal priors with null mean and covariance matrices $\text{Cov}(u_k) = G_k \otimes K_k$ where u_k represent the vector of effects for the kth random effects (sorted by subject first and trait within subject), G_k is an $r \times r$ (within-subject) covariance matrix of the kth random effect and K_k is a user-defined (between subjects) covariance matrix for the kth random effect. For instance, may be a pedigree or marker-based relationship matrix.

Finally, the covariance matrices of random effects are assigned Inverse Whishart priors (for the case of unstructured options, see below) or priors that are structured according to some model (the options implemented are diagonal, recursive and factor analytic).

Algorithm

Internally, MTM uses an orthogonal representation of each of the random effects which is based on the eigenvalue decomposition of the covariance structures (K_k). Samples from all the model unknowns are drawn using a Gibbs sampler (i.e., based on fully conditional distributions).

Parameters

The arguments of the MTM function are:

- `Y` ($n \times q$), a numeric matrix with phenotypes (rows for individuals, columns for traits, `NA`s can be present)
- `XF` ($n \times s$), the design matrix for fixed effects. Should be a numeric matrix with individuals in rows and predictors in columns. The same effects are included for all traits with phenotypes (rows for individuals, columns for traits. `NA`s can be present).
- `K` (q-dimensional list), a hierarchical list where the first level is used to specify random effects (each entry of K defines one random effect). Inside each of the elements of the list a nested list provides the elements needed to specify a random effect. The examples printed below illustrate how to specify various covariance structures.
- `resCov` (list), used to specify the structure of the residual covariance matrix.
- `nIter`, `thin`, `burnIn` (int), the number of iterations, thinning and burn-in for the sampler.
- `saveAt` (character), a path and a prefix for the files generated by MTM (these files contain the samples collected by MTM).
- ...

Examples

In the following examples we illustrate how to specify random and fixed effect and covariance structures in MTM.

Random effects with unstructured covariance matrices

The first random effect will be unstructured and will be assigned a Scaled-Inverse Chi-square with degree of freedom (scalar) `df0`, and scale (matrix, $r \times r$) `S0`. The following example illustrates how to fit a multiple-trait model using a data set available in the BGLR package comprising 599 wheat lines and 4 traits. In this example the covariance matrix of the random effect and that of model residuals are specified as unstructured.

```
rm(list = ls())

library(MTM)
library(BGLR)

data(wheat)
Y <- wheat.Y
A <- wheat.A

fm <- MTM(
  Y = Y,
  K = list(
    list(
      K = A,
      COV = list(
        type = 'UN',
        df0 = 4,
        S0 = diag(4)
      )
    )
  ),
  resCov = list(
    type = 'UN',
    S0 = diag(4),
    df0 = 4
  ),
  nIter = 120,
  burnIn = 20,
  thin = 5,
  saveAt = 'ex1_'
)

# Retrieving estimates
fm$YHat # predicitons
fm$resCov$R # residual covariance matrix
fm$K[[1]]$G # genetic covariance matrix
fm$K[[1]]$U # random effects

# Samples
list.files(pattern = 'ex1_')
plot(scan('ex1_logLik.dat'),type = 'o')
G <- read.table('ex1_G_1.dat')
R <- read.table('ex1_R.dat')
```

Example 1: Random effect model with unstructured covariance matrices

Diagonal covariance matrix

The following example illustrates how to specify a diagonal covariance structure on model residuals. The same specification could be used for the covariance matrix of random effects.

```
rm(list = ls())

library(MTM)
library(BGLR)

data(wheat)
Y <- wheat.Y
A <- wheat.A

fm <- MTM(
  Y = Y,
  K = list(
    list(
      K = A,
      COV = list(
        type = 'UN',
        df0 = 4,
        S0 = diag(4)
      )
    )
  ),
  resCov = list(
    type = 'DIAG',
    S0 = rep(1, 4),
    df0 = rep(1, 4)
  ),
  nIter = 120,
  burnIn = 20,
  thin = 5,
  saveAt = 'ex2_'
)

# Retrieving some estimates
fm$resCov$R # residual covariance matrix

# Samples
R <- read.table('ex2_R.dat')
head(R)
```

Example 2: Diagonal residual covariance matrix

Factor Analytic

In Factor Analysis (FA), a covariance matrix is decomposed into common and specific factors according to $G_k = B_k B_k' + \Psi_k$ where B_k is a matrix of loadings (regressions of the original random effects into common factors) and Ψ_k is a diagonal matrix whose non-null entries give the variances of factors that are trait-specific. The number of common factors (the number of columns of B_k) is specified with the argument `nF`. Restrictions apply as with any FA model on the maximum number of factors that can be specified depending on the number of traits, in particular `r(1 + nF) <= r * (r + 1) / 2`. In MTM the loadings are assigned flat priors (normal priors with null mean and large variance) and the variances of the specific factors are assigned scaled-invers chi-squared with df and scale given by parameters `df0` and `S0` (see example below). The following example modifies Example 2 by specifying a 1-common factor model. The matrix `M` in the example is a logical matrix of the same dimensions as B_k with `TRUE` for loadings that the user want to estimate, and `FALSE` for those that should be zeroed out. The parameter `$var` is the prior variance of the Gaussian prior assigned to the unknown loadings.

```
rm(list = ls())

library(MTM)
library(BGLR)

data(wheat)
Y <- wheat.Y
A <- wheat.A

M <- matrix(nrow = 4, ncol = 1, TRUE)

fm <- MTM(
  Y = Y,
  K = list(
    list(
      K = A,
      COV = list(
        type = 'FA',
        M = M,
        S0 = rep(1, 4),
        df0 = rep(1, 4),
        var = 100
      )
    )
  ),
  resCov = list(
    type = 'DIAG',
    S0 = rep(1, 4),
    df0 = rep(1, 4)
  ),
  nIter = 1200,
  burnIn = 200,
  thin = 5,
  saveAt='ex3_'
)
```

```
# Retrieving some estimates
fm$K[[1]]$B # Estimated loadings
fm$K[[1]]$PSI # Estimated variance of trait-specific factors
fm$K[[1]]$G # Posterior mean of BB'+Psi

# samples
list.files(pattern='ex3_')
```

Example 3: Factor analysis

Recursive models

In recursive models covariances are modeled as regressions. For instance, a recursive model for the covariance matrix of the vector x can be $x = Bx+\delta$ where B is matrix of regression coefficients (typically lower-triangular) with zeros in the diagonal and δ is a random vector (independent normal random variables in MTM). The covariance matrix of x is $\text{Cov}(x, x') = (I - B)^{-1}\Psi((I - B)^{-1})'$ where Ψ is the covariance matrix of δ(diagonal in MTM). The parameters of the model include the recursive effects and the variance of δ. The first ones are assigned IID normal priors with null mean and variance `$var` and the variances are assigned IID scaled-inverse chi-squares priors. The following example illustrates the implementation of a recursive covariance structure in MTM.

```
rm(list = ls())

library(MTM)
library(BGLR)

data(wheat)
Y <- wheat.Y
A <- wheat.A

M <- matrix(nrow = 4, ncol = 4, FALSE)
M[3, 2] <- M[4, 2] <- T # Adding recursion from trait 2 onto traits 3 and 4
M[4, 3] <- T # Adding recursion from trait 3 on trait 4

fm <- MTM(
  Y = Y,
  K = list(
    list(
      K = A,
      COV = list(
        type = 'REC',
        M = M,
        S0 = rep(1, 4),
        df0 = rep(1,4),
        var = 100
      )
    )
  ),
  resCov = list(
    type = 'DIAG',
```

```
    S0 = rep(1, 4),
    df0 = rep(1, 4)
  ),
  nIter = 1200,
  burnIn = 200,
  thin = 5,
  saveAt='ex4_'
)

# Retrieving some estimates
fm$K[[1]]$B # Recursive effects
fm$K[[1]]$PSI # Estimated variance
fm$K[[1]]$G # Posterior mean of the covariance matrix

# It should not be that different than
T = solve(diag(4) - fm$K[[1]]$B)
T %*% diag(fm$K[[1]]$PSI) %*% t(T)

# Samples
list.files(pattern='ex4_')
```

Example 4: Recursive

Adding fixed effects

Fixed effects can be added in any of the examples presented above by adding an argument `XF` which is the design matrix for fixed effects. To illustrate we modify Example 1 by: (i) adding a simulated fixed effect, and (ii) including it when fitting the model.

```
rm(list = ls())

library(MTM)
library(BGLR)

data(wheat)
Y <- wheat.Y
A <- wheat.A

# Simulating a fixed effect
Z <- matrix(rnorm(2*599), ncol = 2)
B <- rbind(c(1, 2, 3, -2), c(-2, 1, 0, 2))
for (i in 1:4) {
  Y[, i] <- Y[, i] + Z %*% B[, i]
}

fm <- MTM(
  XF = Z,
  Y = Y,
  K = list(
    list(
```

```
      K = A,
      COV = list(
        type = 'UN',
        df0 = 4,
        S0 = diag(4)
      )
    )
  ),
  resCov = list(
    type = 'UN',
    S0 = diag(4),
    df0 = 4
  ),
  nIter = 1200,
  burnIn = 200,
  thin = 5,
  saveAt='ex5_'
)

# Retrieving estimates of fixed effects
fm$B.f
```

Example 5: Mixed-effects model

Using eigenvalues and eigenvectors

In the examples presented above, the argument `K` was used to specify covariances between effects of the different the levels of a random effect. Instead, alternatively, one can also provide the eigenvalue decomposition. The following example is equivalent to Example 1 but uses the eigenvalue decomposition of `A`, instead of `A` as input.

```
rm(list = ls())

library(MTM)
library(BGLR)

data(wheat)
Y <- wheat.Y
A <- wheat.A

# Simulating a fixed effect
EVD <- eigen(A)

fm <- MTM(
  Y = Y,
  K = list(
    list(
      EVD = EVD,
      COV = list(
        type='UN',
```

```
        df0 = 4,
        S0 = diag(4)
      )
    )
  ),
  resCov = list(
    type = 'UN',
    S0 = diag(4),
    df0 = 4
  ),
  nIter = 1200,
  burnIn = 200,
  thin = 5,
  saveAt='ex6_'
)
```

Example 6: Using eigenvalues and eigenvectors

Authors

Development & Maintenance

- Gustavo de los Campos (gdeloscampos@gmail.com)
- Alexander Grüneberg (alexander.grueneberg@googlemail.com)

Contributors

- Ana I. Vazquez (anainesvs@gmail.com)
- Hwasoon Kim (hskim28@uab.edu)