Mixed models with R/Multivariate mixed models

Harold Achicanoy

Wednesday, May 13, 2015

Multivariate Mixed Models

Run only in R version 3.1.3

Note: The current version of this file is kept at http://scs.math.yorku.ca/index.php/Mixed_Models_with_R/Multivariate_Mixed_Models.Rmd

We will explore a variety of ways of fitting multivariate mixed models for more than one response variable. As an example, we will use 'mathach' and 'ses' from the high school data set (hs).

With longitudinal data, multivariate mixed models allows fitting 'doubly multivariate' models with missing data.

Snijders and Bosker suggest specifying the lowest level G-side model with G an unconstrained positive-definite matrix, Z_i matrix equal to the identity, and $R = 0$. Thus:

$$V_i = Z_i G Z_i' + R = G$$

This approach cannot work with software that requires a non-zero R matrix. The algorithm for 'lme', for example, use $\Delta = G/\sigma^2$ as the matrix of parameters over which REML maximization takes place.

With 'MCMCglmm', it is possible to fix the the R side by using a point-mass prior but, as far as I understand, σ^2 must be fixed at a positive value.

In the following we will specify the multivariate model through the R side. With 'nlme' we will add a G-side model the R-side model model previously used with 'gls' to fit a one-level multivariate model.

```
# library(spida)
# library(p3d)
# library(nlme)
# data(hs)
# head(hs)
# hs.w <- hs
```

For 'lme' we reshape the data in long form with respect to the variable 'mathach' and 'ses'

```
# names(hs.w) <- sub('^mathach$', 'y.mathach', names(hs.w))
# names(hs.w) <- sub('^ses$',      'y.ses',     names(hs.w))
# head(hs.w)
# dd <- reshape( hs.w, direction = 'long', varying = 2:3, timevar = 'var')
# dd$var <- as.factor(dd$var)
# dd$v <- as.numeric(dd$var)
# dim(hs); dim(dd)
# head(dd)
```

A one-level multivariate model would have the form:

```
# fit.onelevel <- gls( y ~ var/(Sex + Sector) -1,  dd,
#                      correlation =  corSymm( form = ~ v |id), # v must be integer
#                      weights = varIdent(form = ~ 1 | v))
# summary(fit.onelevel)
# getVarCov(fit.onelevel)

# fit.math <- lm( mathach ~ Sex + Sector, hs)
# summary(fit.math)
# summary(fit.math)$sigma^2

# fit.ses <- lm( ses ~ Sex + Sector, hs)
# summary(fit.ses)
# summary(fit.ses)$sigma^2
```

Multilevel Multivariate Model

```
# fit.multilevel <- lme( y ~ var/(Sex * Sector) -1,  dd,
#                      random = ~ var -1| school,
#                      correlation =  corSymm( form = ~ v |school/id),
                        # v must be integer and school
                        # must be included
                        # even though it's not necessary
                        # since 'id' is unique
                        # across schools
#                      weights = varIdent(form = ~ 1 | v))

# summary(fit.multilevel)
# getVarCov(fit.multilevel)
# VarCorr(fit.multilevel)
# plot(fit.multilevel)

# intervals(fit.multilevel)

# wald(fit.multilevel, ":.*:")  # testing for Sex by Sector interaction
```

Why fit a multivariate model?

Two possible reasons: one is interested in estimating the partial correlations among variables where the variables are treated symmetrically.

Another reason is to analyze data in which more than one variable has missing data and in which it is ressonable to analyze the data under the assumption that missingness is MAR. Under MAR, likelihood analysis will yield correct estimates provided all the non-missing data – on which MAR is based – are included in the model.

In this case, the main interest might nevertheless be in estimating the regression of one of the dependent variables on the other. The MLE of the regression coefficient is now the ratio of the MLE of the covariance to the MLE of the variance of the predictor variable.
Producing a reasonable confidence interval for such a parameter is not straightforward. The MCMC approach makes the task easier since posterior sample for the regression parameter can be constructed by simply taking the ratio of the sampled

MCMCglmm

```
# library(MCMCglmm)

# random intercept model
# fit.w.null <- MCMCglmm( cbind(mathach, ses) ~ trait -1 ,
#          random = ~us(trait):school,
                         # 'trait' is reserved keyword to index dependent variables
                         # unstructured covariance matrix
                         # equivalent to random = ~ trait -1 | school
                         #
#          rcov = ~us(trait):units,
                         # Unstructured covariance matrix.
                         #  'units' is reserved
                         # to refer to individual observations
#          data = hs,
#          family = c("gaussian","gaussian"))
# uses default prior
# windows()
# plot(fit.w.null)
# summary(fit.w.null)


# full model
# fit.w.full <- MCMCglmm( cbind(mathach, ses) ~ trait/(Sex * Sector) -1 ,
#          random = ~us(trait):school,
                         # 'trait' is reserved keyword to index dependent variables
                         # unstructured covariance matrix
                         # equivalent to random = ~ trait -1 | school
                         #
#          rcov = ~us(trait):units,
                         # Unstructured covariance matrix. 'units' is reserved
                         # to refer to individual observations
#          data = hs,
#          family = c("gaussian","gaussian"))
# uses default prior
# plot(fit.w.full)
# summary(fit.w.full)

# str(fit.w.full)
# fit.w.full$VCV
# fit.w.full$Z
# fit.w.full$X
# fit.w.full$DIC
# fit.w.null$DIC - fit.w.full$DIC


# plot(fit.w.full$Sol)  # fixed effects
# plot(fit.w.full$VCV)  # G and R

# effectiveSize(fit.w.full$VCV)
# HPDinterval(fit.w.full$VCV)
```

Estimating the regression coefficient

```
# effectiveSize(fit.w.full$VCV[,6]/fit.w.full$VCV[,8])
# plot(fit.w.full$VCV[,6]/fit.w.full$VCV[,8])
# HPDinterval(fit.w.full$VCV[,6]/fit.w.full$VCV[,8])
```

Interpreting changes in DIC

```
# fit.w.full$DIC
# fit.w.null$DIC

# fit.w.null$DIC - fit.w.full$DIC
# cbind( 0:10,1/(1+exp(0:10)))
# cbind( "DIC drop"=0:10,
#        "Appx posterior prob or null model" = round(1/(1+exp(0:10)),5))
```