Tests on multivariate REML using dmm().

Neville Jackson

07 July 2025
For dmm_3.2-1

0.1 Introduction

We want to test multivariate REML estimation using dmeopt="fgls" in dmm(). The code for multivariate *fgls* in *dmm()* uses the conventional approach to multivariate analysis of stacking all traits in succesive blocks in the data vector so that the analysis treats the data vector as if it were all one trait. This requires appropriate blocking of the associated matrices. In the case of "fgls" on the dyadic model, this is not simple; the dyadic model errors have a covariance structure which is specified by a commutation matrix combined with the covriance matrix of fixed model residuals.

0.2 Methods

Make up a small dataset, 4 traits with various levels of correlation between the traits.

```
> wxyz.df
  Id y x   w z
1  1 1 2 2.5 3
2  2 2 3 2.7 2
3  3 3 2 0.6 3
4  4 2 3 2.1 2

> cor(wxyz.df)
           Id          y          x          w          z
Id  1.0000000  0.6324555  0.4472136 -0.4484507 -0.4472136
y   0.6324555  1.0000000  0.0000000 -0.8164966  0.0000000
x   0.4472136  0.0000000  1.0000000  0.5165766 -1.0000000
w  -0.4484507 -0.8164966  0.5165766  1.0000000 -0.5165766
z  -0.4472136  0.0000000 -1.0000000 -0.5165766  1.0000000
```

We can use this to test the multivariate code for sanity.

0.3 Results

The data has no genetic variance component, only VarE(I).

0.3.1 Two uncorrelated traits

Traits y and x have zero correlation.

Solve DME's by OLS

```
> junk <- dmm(wxyz.df,I(cbind(x,y)) ~ 1, components=c("VarE(I)"))
Dyadic mixed model fit for datafile: wxyz.df
  ....
> summary(junk)
Call:
summary.dmm(object = junk)

Coefficients fitted by OLS for fixed effects:

  Trait Estimate StdErr CI95lo CI95hi
1     x      2.5  0.289   1.93   3.07

  Trait Estimate StdErr CI95lo CI95hi
1     y        2  0.408    1.2    2.8


Components partitioned by DME from residual var/covariance after OLS-fixed-effects fit:

        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       x:x    0.333  0.122 0.0948  0.572
```

```
        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       x:y -1.6e-17  0.211 -0.413  0.413

        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       y:x -1.6e-17  0.211 -0.413  0.413

        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       y:y    0.667  0.243   0.19   1.14

>
```

So OLS correctly finds

```
> var(wxyz.df$x)
[1] 0.3333333
> var(wxyz.df$y)
[1] 0.6666667
```

because these are the variance estimates assuming zero correlation of x with y. The estimated covariance of x with y is effectively zero.

Solve DME's by feasable GLS - multivariate case

Using 'dmeopt="fgls"' on the same data gives

```
junk <- dmm(wxyz.df,I(cbind(x,y))  ~ 1, components=c("VarE(I)"),dmeopt="fgls")
..............

summary.dmm(object = junk)

Coefficients fitted by OLS for fixed effects:

  Trait Estimate StdErr CI95lo CI95hi
1     x      2.5  0.289   1.93   3.07

  Trait Estimate StdErr CI95lo CI95hi
1     y        2  0.408    1.2    2.8


Components partitioned by DME from residual var/covariance after OLS-fixed-effects fit:

        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       x:x      0.5   0.63 -0.735   1.73

        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       x:y 0.000206  0.556  -1.09   1.09

        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       y:x 0.000206  0.556  -1.09   1.09

        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       y:y      0.5  0.492 -0.464   1.46

>
```

So the total variance (1.0) is correct, but it is allocated equally to the two traits. Is that right? Well lets look at the variances of y and x separately, and then combined

```
attach(wxyz.df)
> var(y)
[1] 0.6666667
```

```
> var(x)
[1] 0.3333333
> var(c(x,y))
[1] 0.5
```

So it has set each trait's variance to the variance we get if we 'stack' x and y. That may be correct... with only 4 datapoints "fgls" may be unable to 'see' that a different variance for each trait is a better fit. The likelihood surface may be very flat.

Solve DME's by feasable GLS - univariate case

If we model single traits with "fgls" fit we get

```
> junk <- dmm(wxyz.df,x ~ 1, components=c("VarE(I)"),dmeopt="fgls")
Dyadic mixed model fit for datafile: wxyz.df
  ....
> summary(junk)
Call:
summary.dmm(object = junk)

Coefficients fitted by OLS for fixed effects:

  Trait Estimate StdErr CI95lo CI95hi
1     x      2.5  0.289   1.93   3.07


Components partitioned by DME from residual var/covariance after OLS-fixed-effects fit:

        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       x:x    0.333  0.272   -0.2  0.867

>
```

which is correct and the same as obtained with OLS.

OLS will always give the same estimates whether multi-trait or single-trait, whereas FGLS will give different estimates for multitrait vs univariate. That is because FGLS uses the entire multi-trait matrix of covariance of residuals so it adjusts for correlated errors within a trait and for correlated errors across traits. Maybe FGLS should only adjust for within trait error covariances? I dont know?

0.3.2 Swap the order of traits

One test of blocking is to put traits in reverse order

```
junk <- dmm(wxyz.df,I(cbind(y,x))  ~ 1, components=c("VarE(I)"),dmeopt="fgls")
..................
fgls iteration starting siga from ols:
             y:y           y:x x:y       x:x
VarE(I) 0.6666667 -3.204938e-17   0 0.3333333
..................
Round =  100  Stopcrit =  0.5425885
Iteration(fgls) completed - count =  100
Failed to converge (fgls)
>
```

It fails to converge. That will always be a problem with tiny datasets

0.3.3 Correlated traits

We can look at $r = 1$ by putting the same trait in twice

```
junk <- dmm(wxyz.df,I(cbind(x,x))  ~ 1, components=c("VarE(I)"),dmeopt="fgls")
.......................
> summary(junk)
Call:
summary.dmm(object = junk)
```

```
Coefficients fitted by OLS for fixed effects:

  Trait Estimate StdErr CI95lo CI95hi
1     x      2.5  0.289   1.93   3.07

  Trait Estimate StdErr CI95lo CI95hi
1     x      2.5  0.289   1.93   3.07


Components partitioned by DME from residual var/covariance after OLS-fixed-effects fit:

        Traitpair Estimate   StdErr   CI95lo   CI95hi
VarE(I)       x:x 0.000612 3.12e-05 0.000551 0.000673

        Traitpair Estimate   StdErr   CI95lo   CI95hi
VarE(I)       x:x 0.000612 3.12e-05 0.000551 0.000673

        Traitpair Estimate   StdErr   CI95lo   CI95hi
VarE(I)       x:x 0.000612 3.12e-05 0.000551 0.000673

        Traitpair Estimate   StdErr   CI95lo   CI95hi
VarE(I)       x:x 0.000612 3.12e-05 0.000551 0.000673
```

All the estimates are the same, which is correct.

```
> var(c(x,x))
[1] 0.2857143
```

It sets each trait to the stacked variance, and the covariance estimate is correct for a correlation of 1.0.

Lets look at other correlations. Try $r = -1$

```
junk <- dmm(wxyz.df,I(cbind(x,z))  ~ 1, components=c("VarE(I)"),dmeopt="fgls")
...............
        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       x:x      0.5    0.5  -0.48   1.48

        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       x:z     -0.5    0.5  -1.48   0.48

        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       z:x     -0.5    0.5  -1.48   0.48

        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       z:z      0.5    0.5  -0.48   1.48


> var(c(x,z))
[1] 0.2857143
```

So the covariance components do correspond to a -1 correlation. I expected stacking two negatively correlated traits would increase the variance .... it did. Pooling negatively correlated things increases the information content !

Finally lets try $r = 0.5$

```
junk <- dmm(wxyz.df,I(cbind(x,w))  ~ 1, components=c("VarE(I)"),dmeopt="fgls")
............
        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       x:x    0.623  0.866  -1.07   2.32

        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       x:w   -0.745  0.783  -2.28   0.79
```

```
        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       w:x   -0.745  0.783  -2.28   0.79

        Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)       w:w     0.89  0.708 -0.497   2.28


> var(c(x,w))
[1] 0.6083929
```

So we have variance estimate for x nearly twice the correct value, but variance estimate for w approximately correct. The covariance estimate is negative , which is not corret. I would expect positively correlated pooling to achieve less than negatively correlated. If we reverse the order i=of traits , this case again fails to converge.... showing that tiny datasets are not suitable for fgls.

0.3.4 A larger dataset

)

We use the sheep dataset that comes with dmm(), and analyse 2 traits.

```
> data(sheep.df)
> sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
               sexcode=c("M","F"),relmat=c("E","A"),keep=T)

> sheep.fitfgls <- dmm(sheep.mdf, I(cbind(Diam,Bwt)) ~ 1 + Sex + Year ,
                    components=c("VarE(I)","VarG(Ia)"),dmeopt="fgls")

Round =  38  Stopcrit =  0.0004251708
Iteration(fgls) completed - count =  38
Convergence achieved (fgls)
..................
         Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)  Diam:Diam    0.491 0.0608  0.372   0.61
VarG(Ia) Diam:Diam    2.104 0.3198  1.477   2.73

         Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)   Diam:Bwt     2.35    NaN    NaN    NaN
VarG(Ia)  Diam:Bwt     3.93  0.528    2.9   4.97

         Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)   Bwt:Diam     2.35    NaN    NaN    NaN
VarG(Ia)  Bwt:Diam     3.93  0.528    2.9   4.97

         Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)    Bwt:Bwt    11.29  1.172   9.00  13.59
VarG(Ia)   Bwt:Bwt     7.35  0.873   5.64   9.06

\end(verbatim}
That looks entirely reasonable.
The corresponding MINQUE results are
\begin{verbatim}
sheep.fit <- dmm(sheep.mdf, I(cbind(Diam,Bwt)) ~ 1 + Sex + Year ,
             components=c("VarE(I)","VarG(Ia)"))
.........
         Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)  Diam:Diam    0.310  0.212 -0.106  0.726
VarG(Ia) Diam:Diam    0.752  0.187  0.385  1.119

         Traitpair Estimate StdErr  CI95lo CI95hi
VarE(I)   Diam:Bwt     1.49  1.112 -0.6950   3.67
VarG(Ia)  Diam:Bwt     1.86  0.981 -0.0619   3.78
```

```
         Traitpair Estimate StdErr  CI95lo CI95hi
VarE(I)   Bwt:Diam     1.49  1.112 -0.6950   3.67
VarG(Ia)  Bwt:Diam     1.86  0.981 -0.0619   3.78

         Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)    Bwt:Bwt     20.8   5.70   9.62   32.0
VarG(Ia)   Bwt:Bwt      4.6   5.02  -5.24   14.4
```

and the corresponding BCML(bias corrected ML) estimates are

```
sheep.fit.bcml <- dmm(sheep.mdf, I(cbind(Diam,Bwt)) ~ 1 + Sex + Year ,
                  components=c("VarE(I)","VarG(Ia)"),fixedgls=T)
....
Iteration(gls-fixed-effects) completed - count =  7
Convergence achieved
GLS-fixed-effects step completed successfully:
.......
Components partitioned by DME from residual var/covariance after GLS-fixed-effects fit:

         Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)  Diam:Diam    0.124  0.189 -0.246  0.494
VarG(Ia) Diam:Diam    2.308  0.158  1.998  2.618

         Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)   Diam:Bwt     1.18  0.857 -0.502   2.86
VarG(Ia)  Diam:Bwt     2.40  0.718  0.998   3.81

         Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)   Bwt:Diam     1.18  0.857 -0.502   2.86
VarG(Ia)  Bwt:Diam     2.40  0.718  0.998   3.81

         Traitpair Estimate StdErr CI95lo CI95hi
VarE(I)    Bwt:Bwt     17.2   3.76   9.78  24.54
VarG(Ia)   Bwt:Bwt      2.5   3.15  -3.67   8.68
```

All 3 methods (REML, MINQUE, BCML) are in thesame ball park for both traits , and for both components, and for the cross-trait-covariance estimates

0.4 Conclusion

Some of these results for the tiny 4 iem dataset look sensible. Others are way out, but not in a way that indicates problems with the multivariate code..

The results for the larger sheep dataset clinch it. The multivariate "fgls" code is working and keeping the traits indexed correctly through the various stages of blocking of matrices.

Results for univariate "fgls" have been tested against a "known" dataset (see dmmOverview.pdf, Section 5.4). The fgls algorithm is correct, we are only checking the multi-trait code here.

We can conclude that the multivariate "fgls" code seems correct, and the corresponding REML estimates should be valid. Final confirmation awaits finding a small enough test dataset with known REML results.