breedR's multitrait features

UI specifications

Facundo Muñoz

March 2016

> Mathematics is, to a large extent,
> invention of better notations
>
> Richard P. Feynman

Introduction

Guiding principles

- Avoid unexplicit syntaxes such as `(1+x|f)`, `us(id)`, `idh(id)` and alikes
- Do not reinvent the wheel: (re)use proven ideas and code
- Favour established syntaxes and practices
- Keep the interface as simple and intuitive as possible
- Leave some room for innovation

Covariance structures

Table 1 summarizes the possible cross-covariance strucutres of a random effect `u` across traits:

Table 1: Covariance structures

Name	Cov. (3 tr)	ASReml	Notes
Uniform	$\sigma^2 \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$	`u`	same effect for all traits. Not supported by PF90 (reshape)
iid	$\sigma^2 \begin{pmatrix} 1 & \cdot & \cdot \\ \cdot & 1 & \cdot \\ \cdot & \cdot & 1 \end{pmatrix}$	`trait:u`	indep. effects sharing a common var. Not supported by PF90 (reshape)
Exchgble	$\begin{pmatrix} \tau^2 & \sigma^2 & \sigma^2 \\ \sigma^2 & \tau^2 & \sigma^2 \\ \sigma^2 & \sigma^2 & \tau^2 \end{pmatrix}$	`u+trait:u`	shared variance and shared covariance. Not supported by PF90 (reshape)
Indep.	$\begin{pmatrix} \sigma_1^2 & \cdot & \cdot \\ \cdot & \sigma_2^2 & \cdot \\ \cdot & \cdot & \sigma_3^2 \end{pmatrix}$	`idh(trait):u`	indep. effects for each trait. PF90: set initial covars at 0
Full	$\begin{pmatrix} \sigma_1^2 & \sigma_{12} & \sigma_{13} \\ \sigma_{12} & \sigma_2^2 & \sigma_{23} \\ \sigma_{13} & \sigma_{23} & \sigma_3^2 \end{pmatrix}$	`us(trait):u`	full covariance matrix. PF90: default.

General multitrait specification

Case 1 | same latent models

Example: two traits, one site, same fixed effects, additive-genetic random effect

$$Y_1 = X\beta_1 + Zu_1 + \varepsilon_1$$
$$Y_2 = X\beta_2 + Zu_2 + \varepsilon_2,$$

where $(u_1, u_2)' \sim N(0, \Sigma_u \otimes G)$ and $(\varepsilon_1, \varepsilon_2)' \sim N(0, \Sigma \otimes I_n)$.

Σ_u is tipically a fully-parameterized 2×2 matrix, but can be constant ($u_1 = u_2$), diagonal (with etiher one or two parameters), or have a *factor-analytic* structure.

Σ is tipically 2×2 diagonal but can have some structure as well.

breedR prototype 1

```
abv <- breedR_additive_genetic(dat, id = 'self', pedigree = ped)

remlf90(
  fixed    = cbind(Y1, Y2) ~ X,
  random   = ~ block + abv, # varnames or defined effects
  var.ini  = c(block = diag(c(1,1)),
               abv   = 'full',
               resid = c(1,.1,2)),
  data     = dat
)
```

- **structured** random effects (i.e. `genetic` and `spatial`) are **defined beforehand**
  - those named objects are subsequently used in the `random` section of the model specification
  - their `var.ini` specification is gathered together with the rest, separating component definition from inference-related technical arguments
  - all random effects together in one formula
  - all initial values for random effects together
- the multidimensional **response** is specified as a **matrix** (or data frame)
- all fixed and random effects are assumed to have **different values for each trait**
  - **transversal effects** (i.e. *uniform* covariance structure) are not directly supported (ultimately by PROGSF90)
  - a workaround is to reshape the data to long-format (see last section)
- the **cross-covariance** structures of random effects are given in `var.ini`
  - only the **independent** and **full** structures are supported
  - **implicit** specification via a diagonal or full matrices
  - **explicit** specification via keywords `independent` or `full`
  - **matrices** can be specified either by a proper matrix or by a vector representing the lower triangle of the matrix
  - the **dimensions** of the matrices are consistent with the number of traits

- for each random effect, the summary of the fitted model should inform about
  - its covariance model (e.g. `iid`, `additive-genetic`, `AR`, ...)
  - its cross-covariance model wrt to the traits (i.e. either `independent` or `full`)

Trait-specific effects

Case 2 | different models per trait

Example: two traits, one site

$$\begin{aligned} Y_1 &= X\beta_1^{(1)} + bl + op_1\beta_2^{(1)} + && && \varepsilon^{(1)} \\ Y_2 &= X\beta_1^{(2)} + bl + && op_2\beta_2^{(2)} + Zu_2 + && \varepsilon^{(2)}, \end{aligned}$$

- fixed effect X and random effect bl enter in both traits (with trait-wise values)
- op_1 and op_2 are trait-specific *operator* fixed effects
- the additive-genetic random effect u only enters trait 2

breedR prototype 2

```
genetic <- breedR_addgen(dat, id = 'self', pedigree = ped)
remlf90(
  fixed     = cbind(Y1, Y2) ~ X + op1 + op2,
  random    = ~ block + genetic,
  traits    = list(op1 = 'Y1', op2 = 'Y2', genetic = 'Y2')
  data      = dat)
```

- Only **trait-specific effects** need to be declared in a separate argument `traits`
  - if omitted, all effects are assumed to be shared by all traits
- Concerns:
  - the formulas are not completely meaningful by themselves
  - possible alternative: trait-wise formulas. But bulky with many traits.

Grouped random effects and random regressions

Case 3 | Multi-Environment Trials (MET)

Example: one trait, three sites (or years)

- Not really a multi-trait case, but traits on different sites are **often treated as different traits**
  - more limited: no support for *transversal* effects (e.g. a global genetic effect separated from the interaction)
- some fixed effects are **common** (e.g. `gg`) while others are nested within site (e.g. `block`)

- some random effects are **independently nested** within site, with independent variances (e.g. `spatial`)
- some random effects are **nested with cross-covariance** between sites (e.g. `genetic`)
- **residual variance** is heterogeneous (indexed by `site`)
- PROGSF90 can implement this *interaction* by defining **separate effects grouped by site**
  - cross-covariance structure either `independent` or `full`
  - the `iid` cross-covariance structure (see Table 1) can also be supported via a regular interaction (i.e. with a single shared variance parameter)

breedR prototype 3

```
remlf90(
  fixed     = Y ~ site,
  random    = ~ site:block + site:fam,
  var.ini   = list(site:block = 1,    # or 'iid'
                   site:fam = c(1,.5,.5,1,.5,1),  # 'or full' or a SPD matrix
                   site:resid = diag(rep(1, 3))), # 'or 'independent'
  data      = dat)
```

- **Interactions** defined with R's standard **colon**
- Cross-covariance structures **implicit** in the initial specifications
  - `site:block`: regular interaction with one variance
  - `site:fam`: full covariance matrix (spec. as lower triangle)
  - `site:resid`: uncorrelated heterog. residuals grouped by site
- `var.ini` also admits **keywords** `iid`, `full` and `independent` to specify cross-covariance structures while using default initial values
- Omitting `var.ini` equivalent to single-variance interactions and homogeneous residual variances

Random regressions

- For any random effect `B` in the previous setting, `A:B` only makes sense when `A` is a factor
  - meaning: *define separate correlated (or not) effects `B` for each level of `A`*
- it is equivalent to a random regression with respect to indicator variables $\beta_{1j} I_{A=1} + \beta_{2j} I_{A=2} + \cdots$, where the coefficients β_i are jointly normally distributed vectors with as many elements as levels in `B`. This results in a indicator incidence matrix Z_A times a vector of random regressors $(\beta_1', \beta_2', \ldots)'$
- the covariance structure of the random regressors is given by the initial specification and the covariance structure of `B`
  - `iid`: $\sigma_A^2 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \otimes \Sigma_B$
  - `independent`: $\begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{pmatrix} \otimes \Sigma_B$
  - `full`: $\Sigma_A \otimes \Sigma_B$

- Extend the previous notation to the case where `A` is a numeric variable **or a matrix** $\beta_{1i}A^1 + \beta_{2i}A^2 + \cdots$
- For example:
  - Let `A` be the $(n \times p)$ matrix of Legendre polynomials coefficients (up to order $k = p - 1$) evaluated on the n values of a longitudinal variable (`Year`, or any climatic variable)
  - `breedR` can provide functions to facilitate the computation of such a matrix
  - include `random = ~ A:genetic`
- This covers a general specification of **random regressions**, using the same interface for grouped random effects.

A unified interface

Case 4 | Multitrait with reshaping

- reshape the data from *wide* to *long* format:

```
            trait value id
Y1 Y2 X ->      1    Y1  X
                2    Y2  X
```

- use syntax for MET with new factor `trait` (or whatever)
- this accounts for **all covariance structures** from Table 1:
  - uniform: `random = ~ u` **transversal effect** across traits
  - iid, indep or full: `random = ~ trait:u`
    - distinguished by initial value (number, diagonal or full matrices) or keyword (`iid`, `independent`, `full`)
  - exchangeable: `random = ~ u + trait:u`
- also accounts for trait-specific effects by setting to 0 the corresponding values in the dataset
  - possibly with some helper function (e.g. `at()`)
- We get **multitrait**, **trait-specific**, **multisite** and **random regressions** only by implementing the **interactions** feature.
- **Cost**: requires a previous reshaping step and possibly setting some 0s in the dataset
- Don't know the implications on **performance** (memory, speed, acuracy)
- Some models might become **too complicated** (e.g. simultaneously multitrait, multisite, and multiple years)
- Is it still worth to implement the *true* multitrait interfaces?

Pending tasks

- Look at SAS PROC MIXED interface, for lessons to be learned from there.

Conclusions

- Implementing the *true* multitrait specifications can be worthy:
  - More natural workflow
  - Possibly more performant
  - More flexibility for complicated models
  - It does not hurt to have two different ways of fitting the same model
- Planning:
  - Implement a general multitrait interface with basic functionality (i.e. assume `full` cross-covariances in `var.ini` by default, and no trait-specific effects)
  - Allow structured effects into random by defining them previously
  - Implement the generalized interpretation of interactions in `random`
  - Complete the remaining details

Thanks to Catherine Bastien, Véronique Jorge, Leopoldo Sanchez, Vincent Segura, Marie Pegard, Thibaud Chauvin and Mesfin Gebreselassie for their valuable help and fruitful discussion in the morning session we held together at URAGPF INRA.