Exploiter breedR pour l'évaluation génétique d'un dispositif forestier expérimental

Facundo Muñoz
facundo.munoz@cirad.fr
famuvie

Orléans, Sep. 18, 2018

http://famuvie.github.io/breedR/

Démarche de la formation

- Fichier `excercises.R` qui sert de **guide** structurante :
  - consignes
  - code *templates*
- **Diapositives** pour apporter éléments de motivation et discussion
- **Manuels** `breedR` pour creuser

Objectifs opérationnels

1. Analyser un test de **descendance simple** (1 caractère, 1 site)
2. Tenir compte des **effets environnementaux**
3. Ajuster autres **modèles génétiques courants**
4. Analyser **plusieurs caractères** simultanément
5. Estimer l'**interaction GxE** dans des dispositifs multi-site
6. Se servir d'un serveur de calcul pour **paralléliser** les analyses

1. Analyser un test de descendance simple (1 caractère, 1 site)

1. **Estimer** des effets fixes et aléatoires
2. Extraire et **représenter** graphiquement les estimateurs
3. Estimer des **effets additifs** individuelles ou de groupe

---

1. Calculer l'**héritabilité** du dispositif (après-midi)

2. Tenir compte des effets environnementaux

1. Vérifier l'**indépendance** spatiale des résidus
2. Contrôler la variabilité environnementale à l'aide de **modèles spatiaux**

3. Ajuster autres modèles génétiques courants

(explicatif)

1. Estimer des effets génétiques de **compétition**
2. Estimer des effets additifs individuelles à partir d'une matrice d'apparentement génomique (**GBLUP**)
3. Estimer des effets génétiques individuelles de **dominance**

4. Analyser plusieurs caractères simultanément

(explicatif)

5. Estimer l'interaction GxE dans des dispositifs multi-site

6. Se servir d'un serveur de calcul pour paralléliser les analyses

(explicatif, démonstratif)

Statistical analysis of a 'simple' progeny test

Facundo Muñoz
facundo.munoz@cirad.fr
famuvie

Orléans, Sep. 18, 2018

Excercise 1

Fit a progeny model to the globulus dataset with breedR

Use variables `phe_X` as response and `mum` as grouping variable from the `globulus` dataset (included with `breedR`)

```
## Template
fm0 <- remlf90(
  fixed = y ~ 1,  # response + fixed effects
  random = ~ g,   # random effects
  data = D)       # dataset
```


Progeny Tests

Elements of a dataset

- **response** variable y
  - trait of interest, only 1 ('simple' remember?)
- **ancestor** g
  - or genotype, genetic group, family, provenance, or any **grouping** variable.

Globulus dataset

```
knitr::kable(head(globulus), format = "html")
```

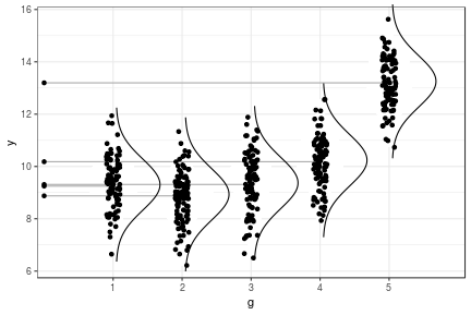
self	dad	mum	gen	gg	bl	phe_X	x	y
69	0	64	1	14	13	15.756	0	0
70	0	41	1	4	13	11.141	3	0
71	0	56	1	14	13	19.258	6	0
72	0	55	1	14	13	4.775	9	0
73	0	22	1	8	13	19.099	12	0
74	0	50	1	14	13	19.258	15	0

Statistical model

We **assume** that the observed value for an individual from group g is

$$y_g \sim \mu_0 + \mu_g + \varepsilon$$

with $\mu_g \sim \mathcal{N}(0, \sigma_g^2)$ and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$



Excercise 2

Include further effects

1. Extend `fm0` by including a **fixed effect of the provenance** (variable gg) and a **random block effect** (variable `bl`).
2. Specify sensible **initial variances** for the random components
3. Extend `fm1` by including an **interaction** between cross-classified variables

Random or fixed?

- Can I change variables from **fixed** to **random** and viceversa?
- How that would change the results and their **interpretation**?

Initial variances

- Good practice: specify **initial values** for all the variances in the model
- `breedR` provides sensible default values, but sometimes you can help

```
fm1 <- remlf90(
  ···,
   var.ini = list(···, resid = ···)
)
```


Interactions

- What is an **interaction effect**?
- When can we estimate interactions?

Nested or crossed variables

A property of the **data**, not the model

- In globulus, the **family** (`mum`) is *nested* within the **provenance** (gg)
- This is a matter of (good) codification:

Nested factors

gg	mum
A	1
A	2
B	3
B	4

Cross-classified factors

gg	mum
A	1
A	2
B	1
B	2

Specifying interactions in breedR

Create a new variable!!

- See `?base::interaction`
- If F1 and F2 are factors

```
F12 <- factor(F1:F2)
```

  is another factor with all the **observed** level combinations

- Use this variable as another **term** in the model

Excercise 3.

Extract results

1. Use functions `summary()`, `fixef()` and `ranef()` to extract **estimates**
2. **Variance estimates** are stored in `fm2$var`
3. **Plot** observed phenotype and `residual()`s vs `fitted()` values to assess the model predictive ability and to diagnose residuals

Excersise 4.

Pedigrees

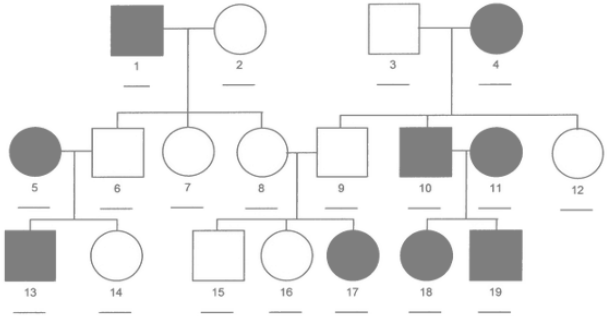
1. Replace the genetic variable `mum` at *family* level, by an *animal model* with a genetic effect at *individual* level.
2. Retrieve the predicted individual breeding values. Make sure they are in the same order as observed in the globulus dataset.

The *animal* model

$$y \sim \mu_0 + a + \varepsilon$$

with $a \sim \mathcal{N}(0, \sigma_a^2 \mathrm{A})$ and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$

where A is the **relationship matrix**, derived from the **pedigree**



Additive Genetic Effect

- Random effect at **individual level**
- Based on a **pedigree**
- BLUP of **Breeding Values** from own and relatives' phenotypes
- Represents the **additive component** of the genetic value
- More **general**:
  - family effect is a particular case
  - accounts for more than one generation
  - mixed relationships
- More **flexible**: allows to select individuals within families

Specification in `breedR`

```
fm3 <- remlf90(
  fixed = ···,
  random = ···,
   genetic = list(
     model = 'add_animal',   # model name for the term
     pedigree = ···,         # pedigree (see Details in ?remlf90)
     id = ···),              # variable name of the individual
  data = globulus
)
```


Specifying a *pedigree*

- A 3-column `data.frame` or `matrix` with the codes for each individual and its parents
- A **family** effect is easily translated into a pedigree:
  - use the **family code** as the identification of a fictitious **mother**
  - use `0` or `NA` as codes for the **unknown fathers**

self	dad	mum
69	0	64
70	0	41
71	0	56
72	0	55
73	0	22
74	0	50

Retrieving BVs from the fitted model

Be careful about the **ordering**

- `ranef(·)$genetic` typically **does not match** the order of observed individuals (founders, clones, recoding)
- Za = `model.matrix(·)$genetic %*% ranef(·)$genetic` always give the PBV for all observed individuals in the dataset, in that order.

Sorting BVs

Observed data:

self	dad	mum
4	1	2
3	1	2

Implied pedigree:

self	dad	mum
1	NA	NA
2	NA	NA
3	1	2
4	1	2

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{model.matrix}(\cdot)\$\text{genetic}} \cdot \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}}_{\text{ranef}(\cdot)\$\text{genetic}} = \begin{bmatrix} a_4 \\ a_3 \end{bmatrix}$$

Excercise 5.

Heritability

Compute the heritability of the globulus dataset using the three available methods.

Methods for computing heritability

Method	Limitations
1. Automatic	`genetic` term method `ai` fixed formula
2. Explicit Formula	method `ai`
3. Bootstrap	-

See: `RShowDoc("Heritability", package = "breedR")`.

Formula syntax

For a single trait model such as:

```
fm <- remlf90(
  fixed = y ~ x1 + x2,
  random = u + v,
  genetic = ···,
  progsf90.options = paste('se_covar_function h2', h2fml),
  data = globulus
```

count the model terms in order as:

term:	x1	x2	u	v	genetic
i:	1	2	3	4	5

And build a formula string **without spaces** as in:

```
h2fml <- 'G_5_5_1_1/(G_5_5_1_1+G_4_4_1_1+R_1_1)'
```

Bootstrap procedure

1. **Fit** the model to your data.
2. Write a function to **simulate observations** from the fitted model. You can use `breedR.sample.phenotype()` if the model is simple enough

```
resample_data <- function(fit) {
  ## Use the estimated values in fit to produce a new data.frame
  ## of the same size and with the same variables as globulus.
  return(dat)
}
```

3: write a function to fit a simulated dataset and extract the target values

```
sim_target() <- function(dat) {
  ## Fit the same model as fm3 to this fake dataset dat
  ## Return the point estimates of all variances and heritability
  return(estimates)
}
```


Resample parameters of interest

4: Replicate the data-generation+estimation process many times

Warning: this can take a while, depending on the model and the dataset.

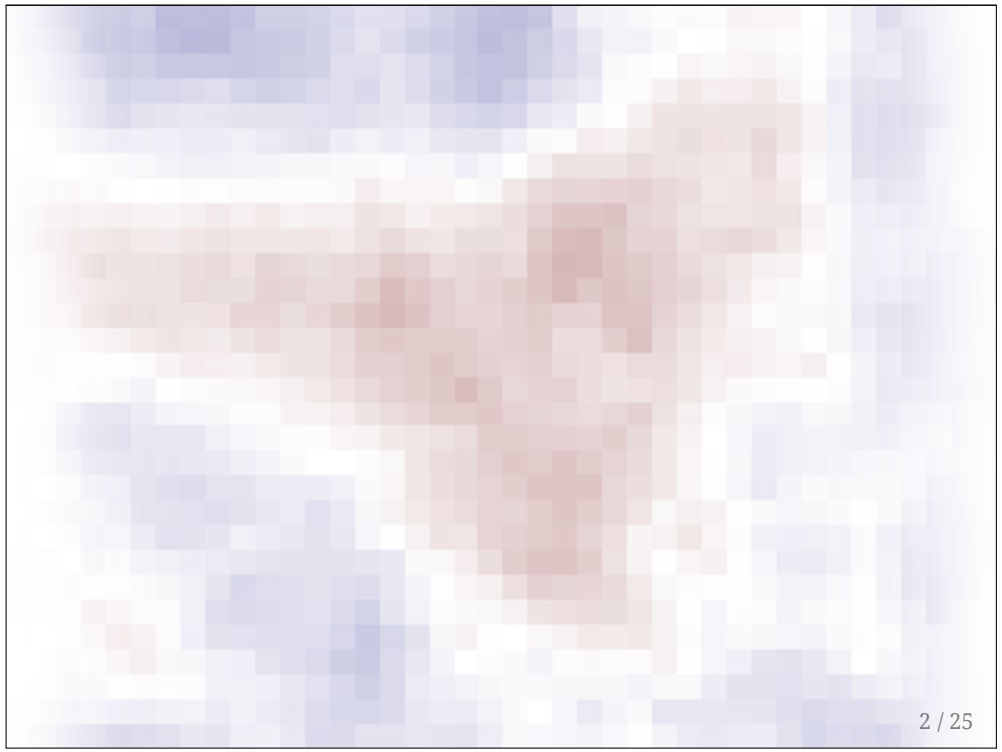
```
boot_estimates <- function(N, fit) {
  ans <- replicate(N, sim_target(dat = resample_globulus(fit)))
  return(as.data.frame(t(ans)))
}

empirical_dist <- boot_estimates(N = 100, fit = fm3)
```


This should account for most analyses of 'simple' progeny tests

Environmental effects

Facundo Muñoz
facundo.munoz@cirad.fr
famuvie

Orléans, Sep. 18, 2018



Excercises 1 and 2

Fit an animal model without the `blocks` effect

- for use as a reference

Asess the spatial independence of residuals by

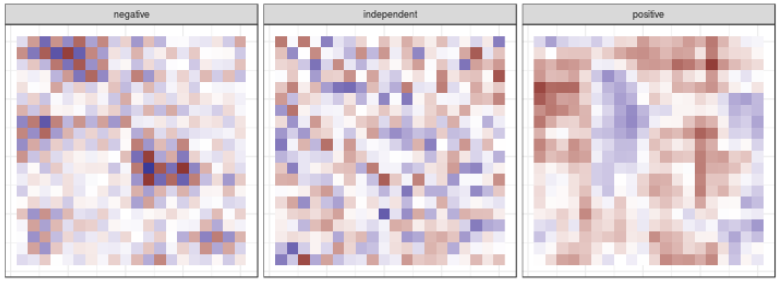
- plotting their spatial distribution
- interpreting the variogram of residuals

Motivation

- **Environmental** sources of variation
- **Bias** genetic estimates
- The residuals of any LMM must be **noise**
- Recommended to **routinely** include spatial effects (Gilmour, Cullis, and Verbyla 1997; Dutkowski et al. 2002)

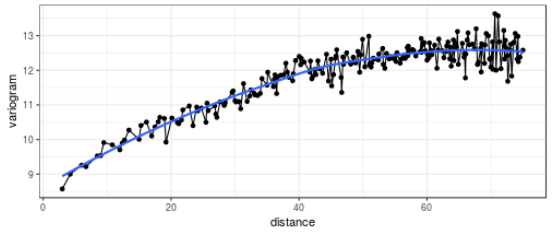
Spatial autocorrelation

observations that are close to each other **tend to be more similar** that observations that are far away (in the positive case)



Empirical (isotropic) semivariogram

$$\gamma(h) = \frac{1}{2} V[Z(\mathrm{u}) - Z(\mathrm{v})], \quad \mathrm{dist}(\mathrm{u}, \mathrm{v}) = h$$



Examining residual autocorrelation in breedR

```
res <- remlf90(···)
```

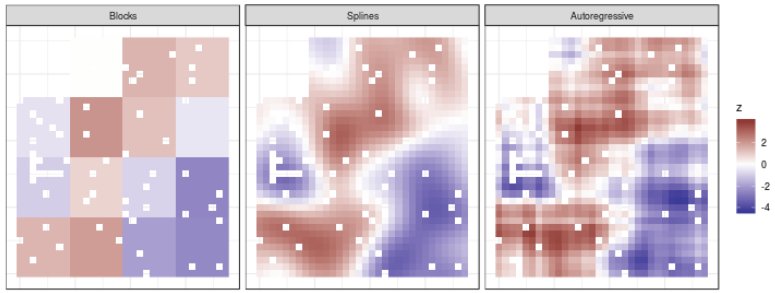
```
plot(res, type = "residuals")
```

```
variogram(res)
```


Excercise 3

Extend `fm4` with each of the three possible spatial models in `breedR`

Spatial models in breedR



Blocks model

$$Zu, \quad u \sim \mathcal{N}(0, \sigma_s^2 I)$$

- u is the vector of random effects for the blocks
- Z is an indicator matrix such that $Z[i, j] = 1$ if the observation i belongs to block j
- σ_s^2 is the spatial variance parameter
- The **block** effect, is a very particular case of spatial effect:
  - It is designed from the begining, possibly using prior knowledge
  - Can account for non-spatial effects (e.g. operator)
  - Introduces **independent** effects between blocks
  - Most neighbours are within the same block (i.e. share the same effect)

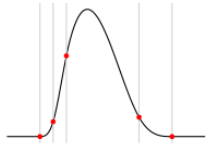
Blocks in breedR

```
fm_bl <- remlf90(
    ...
    spatial = list(
      model = 'blocks',   # spatial model name
      coord = ...,        # matrix or data.frame with coordinates
       id = 'bl'),      # name of the variable
    ...
  )
```

It is equivalent to a `random` effect `bl` (with coordinates)

Splines

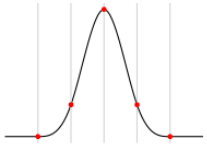
A **cubic B-spline** $B(x)$:



- **Piecewise** curve defined in the intervals determined by 5 **knots**
- Each *piece* is a polynomial of 3rd degree

Splines

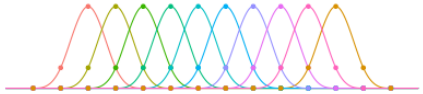
A **cubic B-spline** $B(x)$ with regularly spaced knots:



- The curve is constrained for C^2 **continuity** at each knot
- Only 1 degree of freedom controls the **scale**

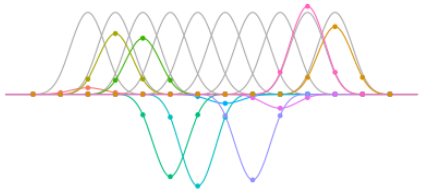
Splines

A number of overlapping curves form a **base** of B-splines $\{B_j(x)\}$



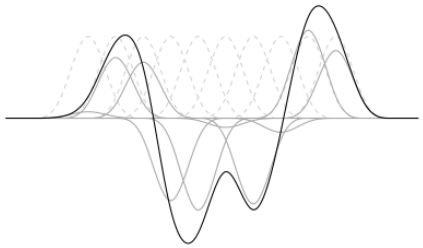
Splines

Each, can be **scaled** using a coefficient $\{u_j B_j(x)\}$



Splines

And **summed** to a **linear combination** $f(x) = \sum_j u_j B_j(x)$



Bidimensional Splines in a Mixed Model

- $f(x) = \sum_j u_j B_j(x)$ provides a **spline representation** of a wide family of curves, in terms of a vector of coefficients u
- For any set of points $x = \{x_i\}$, the vector of values $f(x_i)$ can be written as a matrix operation $f = [B_j(x_i)]u$
- breedR extends this to **two dimensions** and defines a random effect

$$Bu, \quad u \sim \mathcal{N}(0, \sigma_s^2 R_s)$$

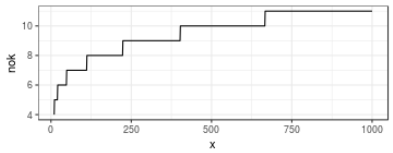
  - u is the vector of spline effects
  - B is the matrix of spline bases evaluated at the observations
  - σ_s^2 is the spatial variance parameter
  - R_s imposes a fixed positive correlation between coefficients of neighbouring spline bases

Splines in breedR

```
fm_sp <- remlf90(
    ...
    spatial = list(
      model   = 'splines',  # spatial model name
      coord   = ...,        # matrix or data.frame with coordinates
      n.knots = c(nk1, nk2)  # N of internal knots in each dim
    ...
  )
```


Number of knots of a `splines` model

- The **smoothness** of the spatial surface can be controlled modifying the number of base functions
- This is directly determined by the **number of knots** (nok) in each dimension
- If not explicitly set, it is determined heuristically by breedR as a function of the number of observations

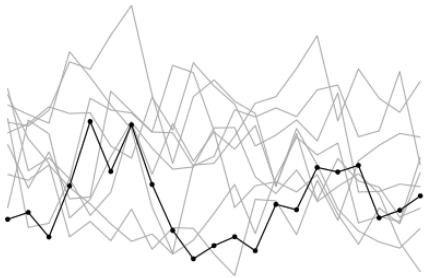


First-Order Autoregressive Process

- An AR1(ρ) on the line is a collection of random variables $\{x_i\}$ where

$$x_t = \rho x_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0,1), |\rho| < 1$$

- A few random simulations with $\rho = 0.5$:



Bidimensional First-Order Autoregressive Process

breedR extends this model to the plane using and defines a component

$$Zu, \quad u \sim \mathcal{N}(0, \sigma_s^2 R_{\text{AR}})$$

- u is the vector of random effects **for each individual location** on a regular grid
- Z is an **indicator matrix** such that $Z[i, j] = 1$ if the observation i is at site j
- σ_s^2 is the spatial variance parameter
- R_{AR} defines a separable correlation structure based on the kronecker product of two AR1 processes

AR in breedR

```
fm_ar <- remlf90(
    ...
    spatial = list(
      model   = 'AR',        # spatial model name
      coord   = ...,         # matrix or data.frame with coordinates
      rho    = c(r1, r2)    # autocorrelation coef in each dim
    ...
  )
```


Autoregressive parameters of a AR model

- The **smoothness** of the `AR` effects can be controlled by the autoregressive parameters (ρ_x, ρ_y) in each dimension
- They can be **given explicitly**
- Otherwise, breedR fits a model for each combination of parameters in a default grid and returns the most likely values

Excercise 4

Plot and compare the predicted spatial effect from each model

You can simply use the `plot()` function with `type = "spatial"` (also `fullspatial`, check the difference).

In order to compare the three plots under the same scale, use `compare.plots(list(p1, p2, p3))`. See `?compare.plots`.

Environmental effects

Further common genetic models

Facundo Muñoz
facundo.munoz@cirad.fr
famuvie

Orléans, Sep. 18, 2018

Competition

a.k.a. Indirect Genetic Effects

Diagnosis of Competition

1. Plot of residuals vs average neighbouring residuals

   Negative correlation, after accounting for Direct Genetic Effects and Spatial Autocorrelation

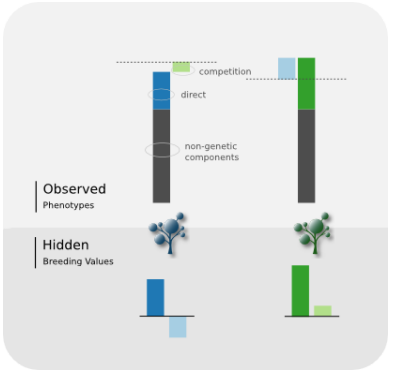
2. Variogram assessment

   Peak at the first lag in the variogram of residuals, after accounting for direct genetic effects and spatial autocorrelation

3. Model comparison

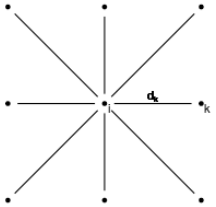
   Compare (*e.g.* AIC) *competition* vs. *animal* models

Competition model



- Each individual have **two** (unknown) Breeding Values (BV):
  - `direct` BV affects its **own** phenotype,
  - `competition` BV affects its **neghbours'**
- The total effect of the neighbouring `competition` BVs is given by their **distance-weighted sum**

Weighted neighbour competition effect



$$\omega_i(\alpha) = \sum_{k \in \partial i} z_{ik}(\alpha) u_{c,k}$$

Where ∂i be the set of **neighbouring locations** of tree i, $u_c = (u_{c,k})'$ the vector of **competition BVs** and $z_{ik}(\alpha) \propto 1/d_{ik}^{\alpha}$, such that

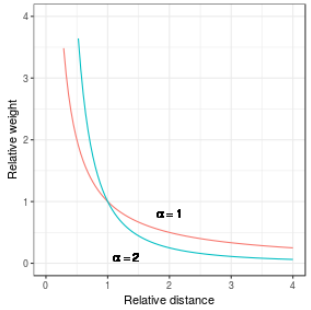
$$\sum_{k \in \partial i} z_{ik}(\alpha)^2 = 1.$$

This condition is **variance-estabilizer** ensuring $\forall i$:

$$\mathrm{Var}(\omega_i) = \mathrm{Var}(u_c) = \sigma_c^2$$

The decay parameter

The **decay parameter** α controls the **relative intensity of competition** of the neighbours



- The weights z_{ik} are **scale-invariant**
- *e.g.* a tree twice as far is weighted $1/2^{\alpha}$ as much
- higher values of α concentrate the weights on the closest trees

Random-effect representation

$$Z_d u_d + Z_c(\alpha) u_c, \quad \begin{pmatrix} u_d \\ u_c \end{pmatrix} \sim \mathcal{N}(0, \Sigma_a \otimes A), \quad \Sigma_a = \begin{pmatrix} \sigma_d^2 & \sigma_{dc} \\ \sigma_{dc} & \sigma_c^2 \end{pmatrix}$$

- Each set of BVs is modelled as a zero-mean **random effect** with structure matrix given by the **pedigree** and independent **variances** σ_d^2 and σ_c^2
- Both random effects are modelled jointly with **covariance** σ_{dc}
- Z_d is an indicator matrix linking observations and individuals
- $Z_c(\alpha)$ weights the competition effect of the neighbours with (fixed) **decay** parameter α

Permanent Environmental Competition Effect

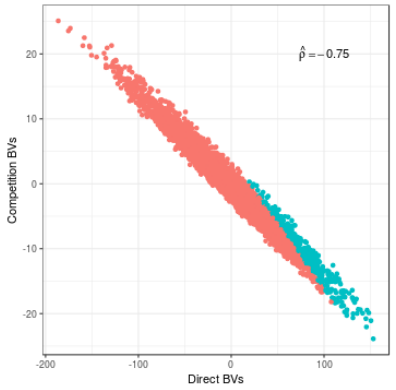
$$Z_p u_p, \quad Z_p = Z_c, u \sim \mathcal{N}(0, \sigma_p^2 I)$$

- **Optional** companion effect with **environmental** (rather than genetic) basis
- Modelled as an individual **independent** random effect that affects **neighbouring** trees in the same (weighted) way

Implementation in breedR

```
fm <- remlf90(
    fixed = ···,
    random = ···,
     genetic = list(
       model = 'competition',
       pedigree = ···,
       coord = ···,
       id = ···,
       pec = TRUE/FALSE),
    data = ···,
     method = 'em'
  )
```


Selection under competition



GBLUP & Dominance effects

Using genomic markers

$$Zu, \quad u \sim \mathcal{N}(0, \sigma^2_G G)$$

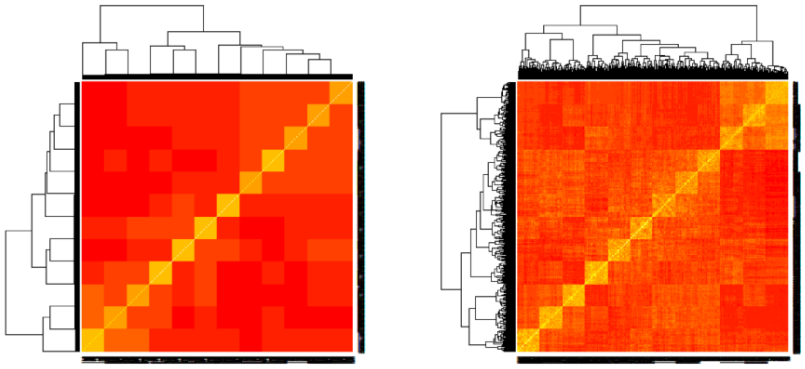
- Use markers to compute a **relationship matrix** G for individuals
  - Several methods available
  - *e.g.* VanRaden et al. 2009

$$G = XX' / \sum 2p(1-p)$$

- **Replace** the `additive-genetic` model, which uses the pedigree-based relationship matrix A with a `generic` model with a genomic relationship matrix G
- Z is an **indicator** matrix linking observations with individuals
- Predicts genetic value of **individuals**, not markers
- Improved **accuracy** wrt pedigree-based evaluation

Relationship matrices

pedigree-based vs. genomic



Note the increased level of detail in the relationship structure

Implementation in breedR

- breedR allows random effects with **arbitrary covariance** structures (`generic` terms, see `?remlf90`)
- These additional components allow to introduce random effects with **arbitrary** incidence and covariance/precision matrices Z and Σ

```
fm <- remlf90(
    ...,
     generic = list(
         G = list(Zg, Gmat),
         ...
         D = list(Zd, precision = Dmat)),
    data = ...
  )
```


Applications

include **additional not-predefined components**

e.g. Dominance, Hybrid populations, Genomic evaluation, etc.

Further common genetic models

- Competition
- GBLUP
- Dominance

Multi-trait models

Facundo Muñoz
facundo.munoz@cirad.fr
famuvie

Orléans, Sep. 18, 2018

Multivariate Linear Mixed Models

2-trait case

$$
\begin{aligned}
Y_1 &= X\beta_1 + Zu_1 + \varepsilon_1 \\
Y_2 &= X\beta_2 + Zu_2 + \varepsilon_2, \\
(u_1, u_2)' &\sim N(0, \Sigma_u \otimes G) \\
(\varepsilon_1, \varepsilon_2)' &\sim N(0, \Sigma \otimes I_n).
\end{aligned}
$$

- Σ_u and Σ either **diagonal** or **fully-parameterized** 2×2 matrices
- Some of the fixed or random effects can affect only a **subset of the traits**
  - *e.g.* fixed effect of operator

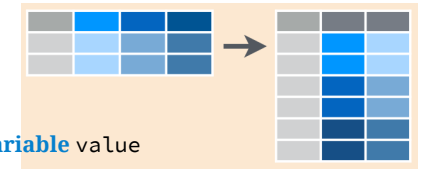
Limitation

of breedR's implementation

- All fixed and random effects are assumed to be **trait-specific**
  - **transversal effects** not directly supported (ultimately by PROGSF90)
- Simpler covariance structures **not supported**
  - *e.g.* independent effects with shared variance, exchangeable structure
- A workaround is to **reshape the dataset** to long-layout

Multi-trait with reshaping

wide to long-layout



- Reshaping operation:
  - Stack traits into a **single variable** `value`
  - Additional variable `trait`
  - Duplicate individual information and other variables
- Use single-trait models with MET syntax
  - `trait` instead of `site`
- This overcomes the limitations breedR's multi-trait implementation
  - more complex models like multi-trait **and** multi-site become cumbersome

Implementation in breedR

Specify the different traits in the main formula using `cbind()`.

```
## Filter site and select relevant variables
dat <-
  droplevels(
    douglas[douglas$site == "s3",
            names(douglas)[!grepl("H0[^4]|AN|BR|site",
                                  names(douglas))]]
  )

res <-
  remlf90(
    fixed = cbind(H04, C13) ~ orig,
    genetic = list(
      model = 'add_animal',
      pedigree = dat[, 1:3],
      id = 'self'),
    data = dat
  )
```


A full covariance matrix across traits is estimated for each random effect, and all results, including heritabilities, are expressed effect-wise:

```
## Formula: cbind(H04, C13) ~ 0 + orig + pedigree
##    Data: dat
##    AIC   BIC logLik
##  30968 31010 -15476
##
## Parameters of special components:
##
##
## Variance components:
##                                          Estimated variances   S.E.
## genetic.direct.H04                                     918.1  438.6
## genetic.direct.H04_genetic.direct.C13                 1872.4  824.0
## genetic.direct.C13                                    5827.6 1829.6
## Residual.H04                                          8373.7  461.7
## Residual.H04_Residual.C13                            10922.0  755.3
## Residual.C13                                         18439.0 1484.2
##
##                   Estimate    S.E.
## Heritability:H04   0.0990 0.04589
## Heritability:C13   0.2391 0.07036
##
## Fixed effects:
##              value    s.e.
## orig.H04.pA 352.00  6.2389
```


Multi-trait models

- Basic multivariate syntax
- Long-shape with `trait` variable

Genotype-Environment interaction

in multi-site trials

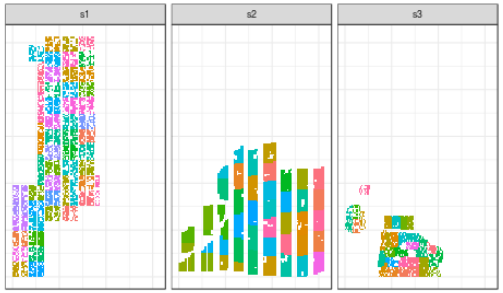
Facundo Muñoz
facundo.munoz@cirad.fr
famuvie

Orléans, Sep. 18, 2018

Excercise 1

Fit a **reference** model for the variable `C13` of the `douglas` dataset (included with `breedR`) with fixed effects of provenance (`orig`) and site (`site`) and a random family (`mum`) effect

Douglas dataset



- Genetic material (provenances and families) planted in 3 sites
- Circumference in 2013 (`C13`) measured in the 3 sites

Excercise 2

Fit a simple **interaction model** with constant variance accross sites.

Multiple *environments*

- **Environments** may refer to sites, but also to years or climates
- Here we will develop the **multi-site** case.
- The other cases might require different approaches (e.g. continuous variation, shared spatial effects, etc.)

Basic interaction

which are the **genotype** and **environment** variables in the `globulus` dataset?

- Remember from module 1 how to crate **interactions**
- Is this interaction a **fixed** or **random** effect?
- what did I mean by *constant variance across sites*?

Basic interaction model

$$
\begin{aligned}
\text{C13} &\sim \text{orig} + \text{site} + \text{mum} + f_e + \varepsilon \\
\text{mum} &\sim \mathcal{N}(0, \sigma^2_f) \\
f_e &\sim \mathcal{N}(0, \sigma^2_{f:e}) \\
\varepsilon &\sim \mathcal{N}(0, \sigma^2)
\end{aligned}
$$

Excercise 3

Extend the previous model to account for **site-varying interaction** variances.

Heterogeneous-variance interaction model

$$\begin{aligned}
\text{C13} &\sim \text{orig} + \text{site} + \text{mum} + \sum_{e=1}^{3} f_e 1_e + \varepsilon \\
\text{mum} &\sim \mathcal{N}(0, \sigma^2_f) \\
f_1 &\sim \mathcal{N}(0, \sigma^2_{f:1}) \\
f_2 &\sim \mathcal{N}(0, \sigma^2_{f:2}) \\
f_3 &\sim \mathcal{N}(0, \sigma^2_{f:3}) \\
\varepsilon &\sim \mathcal{N}(0, \sigma^2)
\end{aligned}$$

Data preparation

- The previous model requires creating 3 new variables to represent the interactions with each site

site	family	f1	f2	f3
s1	1	1		
s1	2	2		
s2	1		1	
s2	2		2	
s3	1			1
s3	2			2

- When added, only one of them will affect the **likelihood** of each observation, depending on its site
- `breedR` will automatically fit three **independent variances**

Excercise 4

Examine and compare the site-specific breeding values from each of the three models above

Extracting Breeding Values

- Remember from Module 1 how to extract PBVs safely (`model.matrix() %*% ranef()`)
- Here, the PBVs are the sum of the main effects and the interactions

Genetic correlations

- Ideally, we would like to fit the following model:

$$
\begin{aligned}
\text{C13} \sim \text{orig} + \text{site} + \text{mum} + \sum_{e=1}^{3} f_e 1_e + \varepsilon \\
\text{mum} \sim \mathcal{N}(0, \sigma_f^2) \\
(f_1, f_2, f_3)' \sim \mathcal{N}\left(0, \Sigma_{G\times E} \otimes \mathrm{I}\right) \\
\varepsilon \sim \mathcal{N}(0, \sigma^2)
\end{aligned}
$$

but unfortunately, `breedR` does not support it.

- Instead, we can always check the **empirical correlations** of the PBVs from the previous model.
- We can also compute the Type-B genetic correlation (as a function of variances)

Ecovalence

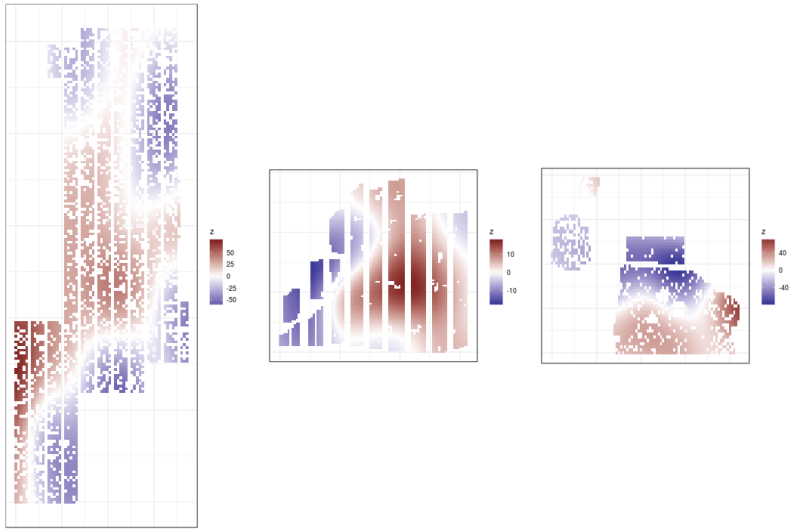
Similarly, we can derive the *Ecovalence* and other measures of interactivity from the PBVs

Site-specific spatial effects

We can leverage the `generic` model to fit three independent spatial effects

```
## Use breedR internal functions to compute splines models on each s
sp1 <- breedR:::breedr_splines(douglas[douglas$site == 's1', c('x', '
sp2 <- ...; sp3 <- ...

## Manually build the full incidence matrices with 0
## and the values computed before
mm1 <- model.matrix(sp1)
inc.sp1 <- Matrix::Matrix(0, nrow = nrow(douglas), ncol = ncol(mm1))
inc.sp1[douglas$site == 's1', ] <- mm1
inc.sp2 <- ...; inc.sp2 <- ...

reml.spl <- remlf90(
  ...
  generic = list(sp1 = list(inc.sp1, breedR:::get_structure(sp1)),
                 sp2 = ...
                 sp3 = ...),
  ...,
  method = 'em'
)
```


Site-specific spatial effects



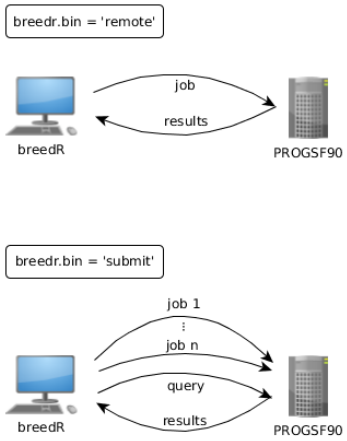
Genotype-Environment interaction

In multi-site trials

Remote parallel computation

Facundo Muñoz
facundo.munoz@cirad.fr
famuvie

Orléans, Sep. 18, 2018



Remote computation

- You have access to a **Linux** server through **SSH**
- You can perform `breedR`'s computations **remotely**
- Take advantage of more **memory** or **faster** processors
- **Parallelize** jobs
- Free **local resources** while fitting models
- See `?remote` for details

Configuring a server

1. Windows users: install `cygwin` with `ssh` beforehand (http://cygwin.org/)
2. configure the client and server machines so that password-less SSH authentication works
3. Set breedR options `remote.host`, `remote.user`, `remote.port` and `remote.bin` (see `?breedR.setOption`)
    - Optionally, set these options permanently in `$HOME/.breedRrc`

```
writeLines(
  c("remote.host = '147.99.222.196'",
    "remote.user = 'yourusername'",
    "remote.bin  = '/usr/local/lib/R/site-library/breedR/bin/'"),
  con = file.path(Sys.getenv('HOME'), '.breedRrc'))
```

Fitting models remotely

```
res <- remlf90(..., breedR.bin = "remote")
```

- Fit model **remotely**
- R-console stays in **stand-by** until job is finished
- When job finishes (provided that connection keeps alive), results are automatically **retrieved**

Identical in use to local computing, but without the processor/memory burden

Submitting jobs

```
res <- remlf90(..., breedR.bin = "submit")
```

- Fit model **remotely**
- Connection is **closed** in the meanwhile
- R-console is **active**
- Typing `res` **queries** the server for the job status (Running/Finished/Aborted)
- **Retrieve** results with `breedR.qget(job_id)`

Parallel computing

- After you **submit** a job, you are free to submit more (specially with multiple-processor servers)
- Query the **status** of all jobs with `breedR.qstat()`
- **Kill** some job with `breedR.qdel(res)` or all jobs with `breedR.qnuke()`

Remote parallel computation