Introduction au modèle linéaire mixte

Timothée Flutre (INRA)

25/06/2018

Résumé

Ce document a pour but de rappeler les bases du modèle linéaire classique, avant de présenter brièvement le modèle linéaire mixte, fondamental en génétique quantitative. L'accent est mis sur l'intuition, et des exemples pratiques sont fournis en R.

Table des matières

1 Préambule

Ce document a été généré à partir d'un fichier texte au format Rmd utilisé avec le logiciel libre R. Pour exporter un tel fichier vers les formats HTML et PDF, installez le paquet rmarkdown (il va vraisemblablement vous être demandé d'installer d'autres paquets), puis ouvrez R et entrez:

```
library(rmarkdown)
render("intro-modlinmix.Rmd", "all")
```

Il est généralement plus simple d'utiliser le logiciel libre RStudio, mais ce n'est pas obligatoire. Pour plus de détails, lisez cette page.

Le format Rmd permet également d'utiliser le language LaTeX pour écrire des équations. Pour en savoir plus, reportez-vous au livre en ligne.

De plus, ce document nécessite de charger des paquets additionnels (ceux-ci doivent être installés au préalable sur votre machine, via `install.packages("pkg")`):

```
suppressPackageStartupMessages(library(MASS))
```

Il est également utile de savoir combien de temps est nécessaire pour exécuter tout le code R de ce document (voir l'annexe):

```
t0 <- proc.time()
```

2 Contexte

Le modèle linéaire mixte étant fondamental en génétique quantitative, ce document peut surtout être d'intérêt pour les étudiants de cette discipline, par exemple ceux suivant l'atelier "Prédiction et Sélection Génomique" organisé et animé par Jacques David et Timothée Flutre depuis 2015. A ce titre, il est recommandé d'avoir déjà lu attentivement le document "Premiers pas" de l'atelier.

Les versions du contenu sont gérées avec le logiciel git, et le dépôt central est hébergé sur GitHub.

3 Le modèle linéaire classique

3.1 Notation

- n: nombre d'observations;
- i: indice indiquant la i-ème observation, donc $i \in \{1, \ldots, n\}$;
- p: nombre de variables explicatives; dans tout ce document, on suppose $n > p$ (mais voir les perspectives à la fin);
- j: indice indiquant la j-ème variable explicative, donc $j \in \{1, \ldots, p\}$;
- $\boldsymbol{y}$: vecteur de dimension n contenant les observations (réponses), modélisées par des variables aléatoires;
- X: matrice d'incidence de dimension $n \times p$ des variables explicatives (prédicteurs), celles-ci pouvant être continues ou discrètes;
- $\boldsymbol{\beta}$: vecteur de dimension p contenant les paramètres correspondant aux effets des variables explicatives sur la moyenne des observations;
- $\boldsymbol{\epsilon}$: vecteur de dimension n contenant les erreurs modélisées par des variables aléatoires;
- R: matrice de dimension $n \times n$ de variance-covariance des erreurs (supposée définie positive, donc inversible); pour commencer, $R = \sigma^2 I$ où I correspond à la matrice identité dont la dimension se déduit du contexte.
- ensemble des données: $\mathcal{D} = \{\boldsymbol{y}, X\}$
- ensemble des paramètres: $\Theta = \{\boldsymbol{\beta}, \sigma^2\}$

3.2 Application

On peut supposer ici qu'on étudie un phénomène tel que le rendement d'une certaine variété de vigne. Notre dispositif expérimental comporte n ceps de cette variété, tous plantés la même année, sur la même variété de porte-greffe, dans la même parcelle qui plus est homogène. L'année des mesures, chaque cep est plus ou moins irrigué (variable explicative $j = 1$) et reçoit plus ou moins d'azote (variable explicative $j = 2$). Sans détailler plus, on suppose que le dispositif expérimental a été "bien pensé" (voir le livre de Dagnelie). Le but de l'expérience est de déterminer l'influence de la variation de ces variables sur la variation du nombre de baies mesuré sur chaque cep au moment de la récolte (variable réponse y_i).

3.3 Vraisemblance

De manière générale, on utilise la lettre majuscule $\mathcal{L}$ pour la vraisemblance, de l'anglais *likelihood*:

$\mathcal{L}(\Theta; \mathcal{D}) = p(\mathcal{D} \mid \Theta)$ où $p()$ indique la densité de probabilité

Dans le cas présent:

$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$ avec $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I) \Leftrightarrow \boldsymbol{y} \mid \boldsymbol{\beta}, \sigma^2 \sim \mathcal{N}(X\boldsymbol{\beta}, \sigma^2 I)$

L'espérance et la variance-covariance des observations sont:

- $\mathrm{E}[\boldsymbol{y} \mid \boldsymbol{\beta}] = X\boldsymbol{\beta}$;
- $\mathrm{Cov}[\boldsymbol{y} \mid \sigma^2] = \mathrm{Cov}[\boldsymbol{\epsilon}] = \sigma^2 I$.

En pratique, on utilise la log-vraisemblance, l:

$l(\Theta; \mathcal{D}) = \log \mathcal{L}(\Theta; \mathcal{D}) = -\frac{n}{2} \log 2\pi - \frac{n}{2} \log \sigma^2 - \frac{1}{2\sigma^2} (\boldsymbol{y} - X\boldsymbol{\beta})^T (\boldsymbol{y} - X\boldsymbol{\beta})$

3.4 Estimation de $\boldsymbol{\beta}$

Géométriquement parlant, les vecteurs $\boldsymbol{\beta}$ candidats appartiennent à $C(X)$, le sous-espace vectoriel engendré par les colonnes de X, c'est-à-dire l'ensemble des combinaisons linéaires des colonnes de X. La longueur du vecteur d'erreurs, $\boldsymbol{\epsilon} = \boldsymbol{y} - X\boldsymbol{\beta}$, est donc la distance euclidienne entre les observations et ce sous-espace.

La méthode des **moindres carrés ordinaires** (*ordinary least squares, OLS*) vise à identifier le vecteur, noté $\hat{\boldsymbol{\beta}}$, qui minimise la somme des carrés des erreurs (*error sum of squares, ESS*):

$\hat{\boldsymbol{\beta}} = \mathrm{argmin}_{\boldsymbol{\beta}}\, ESS$ où $ESS = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - X_i\boldsymbol{\beta})^2$, X_i étant la i-ème ligne de X

En forme matricielle: $ESS = (\boldsymbol{y} - X\boldsymbol{\beta})^T (\boldsymbol{y} - X\boldsymbol{\beta}) = ||\boldsymbol{y} - X\boldsymbol{\beta}||_2^2$, où $||\boldsymbol{v}||_2$ dénote la norme L^2 d'un vecteur $\boldsymbol{v}$ quelconque.

Sa dérivée première vaut $-2X^T(\boldsymbol{y} - X\boldsymbol{\beta})$, et sa dérivée seconde $2X^TX$. Cette dernière étant positive, on en déduit que la fonction à minimiser est convexe et que donc il existe un minimum global, obtenu en égalisant la dérivée première à zéro.

Avec ceci, on retrouve les fameuses **équations normales** de Gauss:

$$X^T X \hat{\boldsymbol{\beta}} = X^T \boldsymbol{y}$$

En indiquant par $^-$ l'opération matricielle d'inversion généralisée (au cas où certaines colonnes de X sont colinéaires), on obtient les estimations:

$$\hat{\boldsymbol{\beta}} = (X^T X)^- X^T \boldsymbol{y}$$

ce qui permet de définir les valeurs ajustées:

$$\hat{\boldsymbol{y}} = X\hat{\boldsymbol{\beta}}$$

et les résidus:

$$\hat{\boldsymbol{\epsilon}} = \boldsymbol{y} - X\hat{\boldsymbol{\beta}} = \boldsymbol{y} - \hat{\boldsymbol{y}}$$

En d'autres termes, parmi tous les éléments de $C(X)$, $\hat{\boldsymbol{\beta}}$ est celui qui minimise la distance euclidienne entre $\boldsymbol{y}$ et $C(X)$. Il est obtenu en projetant orthogonalement $\boldsymbol{y}$ sur $C(X)$, c'est-à-dire de telle sorte que le vecteur de résidus soit orthogonal à toute colonne de X, donc que les produits scalaires soient tous nuls: $\forall j,\ X_j^T(\boldsymbol{y} - X\boldsymbol{\beta}) = 0 \Leftrightarrow X^T(\boldsymbol{y} - X\boldsymbol{\beta}) = 0$. Les valeurs ajustées peuvent donc aussi s'écrire $\hat{\boldsymbol{y}} = P\boldsymbol{y}$ où la matrice de projection orthogonale est $P = X(X^TX)^-X^T$ (on l'appelle souvent la *hat matrix*, notée H).

Notez que la méthode des moindres carrés ne fait pas mention d'une quelconque distribution des erreurs. Mais si l'on suppose la distribution Normale, on peut alors aussi utiliser la méthode du **maximum de vraisemblance** (*maximum likelihood, ML*).

Dans ce cas, on dérive la log-vraisemblance en fonction de $\boldsymbol{\beta}$ qui s'annule en $\hat{\boldsymbol{\beta}}$, et on retrouve les équations normales:

$\frac{\partial l}{\partial \boldsymbol{\beta}}(\hat{\boldsymbol{\beta}}) = 0 \Leftrightarrow (X^TX)\hat{\boldsymbol{\beta}} = X^T\boldsymbol{y}$

En ce qui concerne la quantification de l'incertitude (intervalles de confiance, de crédibilité, etc), veuillez vous référer aux références en fin de document.

3.5 Exemple

Simulation:

```
set.seed(1859)
p <- 3
n <- p * 10^2
X <- matrix(data=rnorm(n=n*p), nrow=n, ncol=p)
(beta <- matrix(rnorm(n=p, mean=0, sd=1)))
```

```
##         [,1]
## [1,]  0.2454
## [2,] -1.2643
## [3,]  0.0606
```

```
sigma2 <- 1
V <- diag(n)
R <- sigma2 * V
epsilon <- mvrnorm(n=1, mu=rep(0,n), Sigma=R)
y <- X %*% beta + epsilon
dat <- as.data.frame(cbind(y, X))
colnames(dat) <- c("y", paste0("X", 1:p))
```

Estimation via les équations normales:

```
(beta.hat <- solve(t(X) %*% X) %*% t(X) %*% y)
```

```
##          [,1]
## [1,]  0.15062
## [2,] -1.29622
## [3,]  0.00554
```

```r
fit <- lm(formula=y ~ -1 + X1 + X2 + X3, data=dat)
matrix(fit$coefficients)
```

```
##          [,1]
## [1,]  0.15062
## [2,] -1.29622
## [3,]  0.00554
```

3.6 Estimation de σ^2

Par définition: $\forall i,\ \sigma^2 = \text{Var}[\epsilon_i] = \text{E}[\epsilon_i^2]$. Il est donc naturel de chercher l'estimateur de σ^2 à partir de l'espérance de la somme des carrés résiduelle (*residual sum of squares, RSS*):

$RSS = \sum_i \hat{\epsilon}_i^2 = (\boldsymbol{y} - \hat{\boldsymbol{y}})^T(\boldsymbol{y} - \hat{\boldsymbol{y}}) = \boldsymbol{y}^T(I - P)\boldsymbol{y}$

où l'on a utilisé le fait que la matrice P est idempotente ($P^2 = P$) et symmétrique ($P^T = P$).

La matrice RSS étant de dimension 1×1, elle est donc égale à sa trace, tr:

$$\begin{aligned}
\text{E}[RSS] &= \text{E}[\text{tr}[\boldsymbol{y}^T(I - P)\boldsymbol{y}]] \\
&= \text{E}[\text{tr}[(I - P)\boldsymbol{y}\boldsymbol{y}^T]] \\
&= \text{tr}[(I - P)\text{E}[\boldsymbol{y}\boldsymbol{y}^T]] \\
&= \text{tr}[(I - P)(X\boldsymbol{\beta}\boldsymbol{\beta}^T X^T + \sigma^2 I)] \\
&= \sigma^2 \text{tr}[(I - P)] \\
&= \sigma^2(n - r(X))
\end{aligned}$$

où on a utilisé les propriétés de commutativité de la trace, et de la trace avec l'espérance, ainsi que $PX = X$, et que la trace d'un projecteur orthogonal est la dimension de l'espace sur lequel il projette, soit ici le rang de X, noté $r(X)$.

Avec les moindres carrés, on obtient donc un estimateur sans biais de σ^2:

$$S^2 = \frac{RSS}{n - r(X)}$$

où $r(X) = p$ quand X est de plein rang.

De la même manière que pour $\boldsymbol{\beta}$, on peut également utiliser le maximum de vraisemblance:

$\frac{\partial \mathcal{L}}{\partial \sigma^2}(\hat{\sigma}^2) = 0 \Leftrightarrow \hat{\sigma}^2 = \frac{1}{n}(\boldsymbol{y} - X\hat{\boldsymbol{\beta}})^T(\boldsymbol{y} - X\hat{\boldsymbol{\beta}}) = \frac{RSS}{n}$

mais celui-ci est biaisé:

$\text{E}[\hat{\sigma}^2] = \frac{n - r(X)}{n}\sigma^2$

Ce biais vient du fait que la perte de degrés de liberté due à l'estimation de $\boldsymbol{\beta}$ est négligée. En d'autres termes, l'estimateur du maximum de vraisemblance utilise $\hat{\boldsymbol{\beta}}$ mais sans prendre en compte son incertitude.

Notez aussi la valeur maximum de la log-vraisemblance: $l(\hat{\Theta}; \mathcal{D}) = -\frac{n}{2}\log 2\pi - \frac{n}{2}\log\hat{\sigma}^2 - \frac{1}{2\hat{\sigma}^2}RSS = -\frac{n}{2}\log\frac{RSS}{n} + \text{constante}$. Ceci permet de faire le lien entre les moindres carrés et le maximum de vraisemblance pour la comparaison de modèles via le **critère d'information d'Akaike**: $AIC = 2 \times \text{nb de paramètres identifiables} + n\log\frac{RSS}{n} + \text{constante}$.

3.7 Exemple (suite)

```
y.hat <- X %*% beta.hat
epsilon.hat <- y - y.hat
(RSS <- t(epsilon.hat) %*% epsilon.hat)
```

```
##      [,1]
## [1,]  264
```

```
rX <- p
(S2 <- RSS / (n - rX))
```

```
##       [,1]
## [1,] 0.889
```

```
(sigma2.hat <- RSS / n)
```

```
##      [,1]
## [1,] 0.88
```

```
fit$rank
```

```
## [1] 3
```

```
(summary(fit)$sigma)^2
```

```
## [1] 0.889
```

```
anova(fit)
```

```
## Analysis of Variance Table
##
## Response: y
##            Df Sum Sq Mean Sq F value Pr(>F)
## X1          1      8       8    9.07 0.0028 **
## X2          1    524     524  589.31 <2e-16 ***
## X3          1      0       0    0.01 0.9162
## Residuals 297    264       1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.8 Erreurs corrélées et/ou avec différentes variances

La matrice de variance-covariance des erreurs, R, peut aussi s'écrire $R = \sigma^2 V$, où V est supposée connue et peut prendre n'importe quelle forme (différentes valeurs sur la diagonale, valeurs non-nulles hors de la diagonale, etc), tant qu'elle est définie positive.

Dans de tels cas, la méthode des **moindres carrés généralisés** (*generalized least squares, GLS*) vise à identifier le vecteur $\hat{\boldsymbol{\beta}}$ qui minimise la somme pondérée des erreurs:

$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} (\boldsymbol{y} - X\boldsymbol{\beta})^T R^{-1} (\boldsymbol{y} - X\boldsymbol{\beta})$

Ceci correspond à minimiser la distance de Mahalanobis du vecteur des erreurs, ce qui aboutit à la solution suivante, très similaire de la précédente:

$\hat{\boldsymbol{\beta}} = (X^T R^{-1} X)^- X^T R^{-1} \boldsymbol{y}$

Le même résultat est obtenu par maximum de vraisemblance.

Le paquet R MASS implémente cette méthode dans sa fonction `gls()`.

4 Le modèle linéaire mixte

Dans le modèle linéaire classique, on ne cherche à modéliser que la moyenne des observations via une fonction linéaire des paramètres: $\mathrm{E}[\boldsymbol{y}|\boldsymbol{\beta}] = X\boldsymbol{\beta}$.

Mais on peut aussi vouloir modéliser leur variance-covariance. Dans ce cas, une manière de faire est d'introduire des vecteurs de variables aléatoires, $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, etc, pour structurer la variance-covariance des observations via une fonction linéaire de composantes de la variance, $\sigma^2_{u_1}$, $\sigma^2_{u_2}$, etc.

Dans la suite, on ne va utiliser qu'un seul vecteur $\boldsymbol{u}$ avec sa composante σ^2_u. Pour le cas plus général, voir les références en fin de document.

4.1 Notations

Les mêmes que précédemment, plus celles-ci:

- q: nombre de variables aléatoires pour structurer la variance-covariance des observations, avec $n > q$;
- k: indice indiquant la k-ème variable aléatoire, donc $k \in \{1, \ldots, q\}$;
- Z: matrice d'incidence de dimension $n \times q$ reliant les y_i aux u_k;
- G: matrice de variance-covariance de dimension $q \times q$ du vecteur $\boldsymbol{u}$, telle que $G = \sigma^2_u A$ où A est connue et définie positive;
- $\boldsymbol{\phi}$: vecteur des composantes de la variance, ici égal à $(\sigma^2_u, \sigma^2)^T$;
- H: matrice de variance-covariance de dimension $n \times n$ dépendant de $\boldsymbol{\phi}$.

Dans le cas du modèle mixte, on nomme souvent $\boldsymbol{\beta}$ les **effets fixes** et $\boldsymbol{u}$ les **variables aléatoires**, d'où le qualificatif "mixte".

4.2 Application

En génétique quantitative, il est fréquent d'analyser un échantillon de q individus différents, phénotypés plusieurs fois chacun, par exemple p années, pour un total de $n = q \times p$ observations.

Lorsque la généalogie (pédigrée) de l'échantillon est connue, on peut l'utiliser pour calculer les relations génétiques attendues entre individus. Ces relations génétiques prennent la forme de la matrice de variance-covariance A de dimension $q \times q$.

4.3 Vraisemblance

$\boldsymbol{y} = X\boldsymbol{\beta} + Z\boldsymbol{u} + \boldsymbol{\epsilon}$ avec $\boldsymbol{u} \sim \mathcal{N}(0, \sigma^2_u A)$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 V)$ et $\mathrm{Cov}[\boldsymbol{u}, \boldsymbol{\epsilon}] = 0$

On peut donc écrire:

$\boldsymbol{y} \mid \boldsymbol{\beta}, \boldsymbol{u}, \sigma^2 \ \sim \ \mathcal{N}(X\boldsymbol{\beta} + Z\boldsymbol{u},\ \sigma^2 V)$

Après intégration des u_k (on dit aussi qu'elles ont été "marginalisées"), on obtient:

$\boldsymbol{y} \mid \boldsymbol{\beta}, \sigma^2_u, \sigma^2 \ \sim \ \mathcal{N}(X\boldsymbol{\beta},\ \sigma^2_u Z A Z^T + \sigma^2 V)$

L'espérance et la variance-covariance des observations sont bien des fonctions linéaires de paramètres:

- $\mathrm{E}[\boldsymbol{y} \mid \boldsymbol{\beta}] = X\boldsymbol{\beta}$;
- $\mathrm{Cov}[\boldsymbol{y} \mid \boldsymbol{\phi}] = H = ZGZ^T + R = \sigma^2_u Z A Z^T + \sigma^2 V$.

4.4 Estimation de β et prédiction de u

En supposant que les composantes de la variance sont connues, il existe plusieurs façons d'obtenir l'estimateur de $\boldsymbol{\beta}$ et le prédicteur de $\boldsymbol{u}$.

La première est plutôt fondée sur le paradigme fréquentiste selon lequel la probabilité d'un évènement est sa fréquence. Pour toute variable inconnue $\boldsymbol{v}$, on cherche la combinaison linéaire des données ($\hat{\boldsymbol{v}}$), ayant la plus petite erreur quadratique moyenne ($\mathrm{E}[(\hat{\boldsymbol{v}} - \boldsymbol{v})^2]$), et étant non-biaisée ($\mathrm{E}[\hat{\boldsymbol{v}}] = \mathrm{E}[\boldsymbol{v}]$). Pour les paramètres $\boldsymbol{\beta}$, on parle de meilleur estimateur linéaire non biaisé (*best linear unbiased estimator, BLUE*) et, pour les $\boldsymbol{u}$, de meilleur prédicteur linéaire non biaisé (*best linear unbiased predictor, BLUP*). (Le terme "prédicteur" vient du fait que, dans le paradigme fréquentiste, les u_k ne sont pas considérés comme des paramètres, le nombre de ces derniers ne devant pas pouvoir augmenter avec la taille de l'échantillon.)

La seconde se réfère plutôt au paradigme bayésien selon lequel la probabilité d'un évènement pour un individu donné est le degré de plausibilité, pour cet individu, de l'occurrence de l'évènement (voir l'article de Robert en 2001). Cette façon permet d'obtenir les mêmes formules que la précédente, bien qu'avec une interprétation différente, mais elle me semble plus intuitive. Dans ce paradigme, il n'y a pas non plus de terminologie différente pour distinguer les différentes variables inconnues (β versus u).

Selon la démarche bayésienne, de manière générale, on commence par écrire la distribution de probabilité conjointe de toutes les quantités, observables ou non, qui se décompose en vraisemblance et distribution *a priori* (le **prior**):

$p(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{u}) = p(\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{u}) \times p(\boldsymbol{\beta}, \boldsymbol{u})$

Pour ces deux termes, on choisit ensuite ce qui semble le plus cohérent en fonction des connaissances disponibles et du processus de collecte des données. Dans le cas du modèle linéaire mixte, pour obtenir les mêmes formules que le BLUP issues du paradigme fréquentiste, ceci équivaut à:

- vraisemblance: $\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{u} \sim \mathcal{N}(X\boldsymbol{\beta} + Z\boldsymbol{u}, R)$ avec $R = \sigma^2 V$;
- prior: on suppose l'indépendance $p(\boldsymbol{\beta}, \boldsymbol{u}) = p(\boldsymbol{\beta}) \times p(\boldsymbol{u})$, avec $\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{0}, S)$ où $S = \sigma^2_\beta I$ et $\sigma^2_\beta \to +\infty$ (donc $S^{-1} \to 0$), ce qui équivaut à une distribution uniforme, et $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, G)$ avec $G = \sigma^2_u A$.

Grâce à la formule de Bayes, l'inférence se réalise alors conditionnellement aux données observées, dans le but de calculer la distribution *a posteriori* (le **posterior**), qui est proportionnelle à la vraisemblance et au prior:

$p(\boldsymbol{\beta}, \boldsymbol{u}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{u}) \times p(\boldsymbol{\beta}, \boldsymbol{u})$

Dans le modèle linéaire mixte, la vraisemblance et le prior sont Normales, donc le posterior l'est aussi, et donc ses moyenne, médiane et mode sont égaux. Pour trouver cette valeur, il suffit de calculer la dérivée première du posterior et de l'annuler:

- $\frac{\partial p(\boldsymbol{\beta}, \boldsymbol{u}|\boldsymbol{y})}{\partial \boldsymbol{\beta}}(\hat{\boldsymbol{\beta}}) = 0$;
- $\frac{\partial p(\boldsymbol{\beta}, \boldsymbol{u}|\boldsymbol{y})}{\partial \boldsymbol{u}}(\hat{\boldsymbol{u}}) = 0$.

Avec ceci, on retrouve les fameuses **équations du modèle mixte** (*mixed model equations, MME*) de Henderson:

$$\begin{bmatrix} X^T R^{-1} X & X^T R^{-1} Z \\ Z^T R^{-1} X & Z^T R^{-1} Z + G^{-1} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\beta}} \\ \hat{\boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} X^T R^{-1} \boldsymbol{y} \\ Z^T R^{-1} \boldsymbol{y} \end{bmatrix}$$

Ces équations ont l'intérêt de ne pas faire intervenir H, de grande dimension $n \times n$, donc coûteuse à inverser, mais R et G, toutes les deux de plus petites dimensions et de structures souvent plus simples (matrices creuses). Elles sont donc utilisées en pratique pour calculer $\hat{\boldsymbol{\beta}}$ et $\hat{\boldsymbol{u}}$.

En développant ces équations, on obtient:

- $X^T R^{-1} X \hat{\boldsymbol{\beta}} + X^T R^{-1} Z \hat{\boldsymbol{u}} = X^T R^{-1} \boldsymbol{y}$;
- $(Z^T R^{-1} Z + G^{-1}) \hat{\boldsymbol{u}} = Z^T R^{-1} (\boldsymbol{y} - X\hat{\boldsymbol{\beta}})$.

En insérant dans la première équation la valeur de $\hat{\boldsymbol{u}}$ issue de la deuxième, on obtient:

$$\begin{aligned} X^T(R^{-1} - R^{-1}Z(Z^TR^{-1}Z + G^{-1})^{-1}Z^TR^{-1})X\hat{\boldsymbol{\beta}} &= X^T(R^{-1} - R^{-1}Z(Z^TR^{-1}Z + G^{-1})^{-1}Z^TR^{-1})\boldsymbol{y} \\ X^T(Z^TGZ + R)^{-1}X\hat{\boldsymbol{\beta}} &= X^T(Z^TGZ + R)^{-1}\boldsymbol{y} \\ X^TH^{-1}X\hat{\boldsymbol{\beta}} &= X^TH^{-1}\boldsymbol{y} \\ \hat{\boldsymbol{\beta}} &= (X^TH^{-1}X)^-X^TH^{-1}\boldsymbol{y} \end{aligned}$$

qui correspond aux moindres carrés généralisés et au *BLUE*.

De même:

$$\begin{aligned} \hat{\boldsymbol{u}} &= (Z^TR^{-1}Z + G^{-1})^{-1}Z^TR^{-1}(\boldsymbol{y} - X\hat{\boldsymbol{\beta}}) \\ &= GZ^TH^{-1}(\boldsymbol{y} - X\hat{\boldsymbol{\beta}}) \end{aligned}$$

qui correspond au *BLUP*.

En pratique, on remplace les inconnues σ_u^2 et σ^2 par leurs estimations: on dit que l'on calcule les BLUE et BLUP "empiriques".

4.5 Exemple

Simulation:

```
set.seed(1859)
p <- 3
q <- 5
n <- q * 10^2
X <- matrix(data=rnorm(n=n * p), nrow=n, ncol=p)
beta <- matrix(rnorm(n=p, mean=0, sd=1))
Z <- model.matrix(~ -1 + rep(letters[1:q], each=10^2))
dimnames(Z) <- NULL
sigma.u2 <- 1
A <- diag(q)
G <- sigma.u2 * A
u <- matrix(mvrnorm(n=1, mu=rep(0,q), Sigma=G))
sigma2 <- 1
V <- diag(n)
R <- sigma2 * V
epsilon <- mvrnorm(n=1, mu=rep(0,n), Sigma=R)
y <- X %*% beta + Z %*% u + epsilon
```

Estimation/prédiction via les *MME* (simplifiées au cas $G = \sigma_u^2 A$ et $R = \sigma^2 I$):

```
Ainv <- solve(A)
lambda <- sigma2 / sigma.u2
lhs <- matrix(data=NA, nrow=p+q, ncol=p+q) # pour "left-hand side"
lhs[1:p, 1:p] <- crossprod(X, X)
lhs[(p+1):(p+q), 1:p] <- crossprod(Z, X)
lhs[1:p, (p+1):(p+q)] <- crossprod(X, Z)
lhs[(p+1):(p+q), (p+1):(p+q)] <- crossprod(Z, Z) + lambda * Ainv
```

```r
rhs <- matrix(data=NA, nrow=p+q, ncol=1) # pour "right-hand side"
rhs[1:p, 1] <- crossprod(X, y)
rhs[(p+1):(p+q), 1] <- crossprod(Z, y)
solutions <- solve(lhs, rhs)
cbind(truth=c(beta, u), mme=solutions[,1])
```

```
##       truth     mme
## [1,] -0.760 -0.7170
## [2,]  0.468  0.4301
## [3,]  0.248  0.1991
## [4,]  1.101  1.2061
## [5,]  0.560  0.5644
## [6,] -0.241 -0.0611
## [7,]  0.615  0.8065
## [8,] -0.679 -0.5889
```

4.6 Estimation de σ_u^2 et σ^2

Comme dans le cas du modèle linéaire classique, la méthode du maximum de vraisemblance appliquée au modèle linéaire mixte produit des estimateurs biaisés de σ_u^2 et σ^2.

Dans le cadre fréquentiste, une méthode, appelée **maximum de vraisemblance restreinte** (*restricted maximum likelihood, ReML*), a donc été développée spécifiquement pour estimer les composantes de la variance. Elle décompose la vraisemblance en deux parties, dont l'une ne dépend que des variables aléatoires $\boldsymbol{u}$ sans les paramètres $\boldsymbol{\beta}$.

Pour que la vraisemblance ne dépende plus de $\boldsymbol{\beta}$, on cherche des vecteurs tels que $\boldsymbol{v}^T X = 0$. Au maximum, il existe $n - r(X)$ tels vecteurs linéairement indépendants. Si on les range dans une matrice S, on aura $SX = 0$, et donc $\mathrm{E}[S\boldsymbol{y}] = 0$. Ceci est le cas par exemple avec $S = I - X(X^T X)^- X^T$. De plus, on peut montrer que $S = KK^T$ avec $K^T K = I$.

L'inférence se réalise donc via la vraisemblance restreinte des composantes de la variance: $K^T \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{0}, K^T H(\boldsymbol{\phi}) K)$. Une fois cette partie maximisée, on se sert alors de l'estimation obtenue de $H(\hat{\boldsymbol{\phi}})$ pour calculer $\hat{\boldsymbol{\beta}}$ via les moindres carrés généralisés.

Par contre, les estimateurs de cette méthode pour les composantes de la variance ne sont pas disponibles sous forme analytique. Il faut donc recourir à un algorithme itératif, tel que la méthode de Newton ou bien l'algorithme EM (ci-dessous).

Notez aussi que, dans le cadre bayésien, l'équation du ReML est obtenue en utilisant un prior uniforme sur $\boldsymbol{\beta}$ pour les intégrer.

4.7 Algorithme EM

Cet algorithme est très utilisé (bien au-delà des modèles mixtes) car il permet d'obtenir les estimations du maximum de vraisemblance dans des modèles pour lesquels on peut utiliser la notion de "données manquantes" (on parle aussi de "variables cachées" ou "latentes").

Pour l'utiliser, on introduit donc:

— le vecteur des données manquantes, souvent noté $\boldsymbol{z}$, et ici égal à $(\boldsymbol{\beta}^T, \boldsymbol{u}^T)^T$;
— et le vecteur des "données complètes" (on parle aussi de "données augmentées"), noté $\boldsymbol{x}$, et ici égal à $(\boldsymbol{y}^T, \boldsymbol{z}^T)^T$.

Au coeur de l'algorithme EM se trouve une **fonction d'objectif**, souvent notée Q, qui se trouve être l'espérance de la log-vraisemblance des données complètes, $l(\boldsymbol{\phi}; \boldsymbol{x})$, conditionnellement aux données manquantes, $\boldsymbol{z}$, sachant les données observées, $\boldsymbol{y}$, et les paramètres, $\boldsymbol{\phi}$, à la t-ème itération:

$$Q(\boldsymbol{\phi}; \boldsymbol{\phi}^{(t)}) = E_{\boldsymbol{z}|\boldsymbol{y},\boldsymbol{\phi}^{(t)}}[\, l(\boldsymbol{\phi}; \boldsymbol{x}) \mid \boldsymbol{z}|\boldsymbol{y}, \boldsymbol{\phi}^{(t)}\,]$$

L'algorithme EM tient son nom du fait que:

- la première étape (E pour *expectation*) consiste à calculer Q sachant $\boldsymbol{\phi}^{(t)}$;
- et la deuxième étape (M pour *maximization*) consiste à maximiser Q par rapport aux paramètres $\boldsymbol{\phi}$: $\boldsymbol{\phi}^{t+1} = \text{argmax}_{\boldsymbol{\phi}}\, Q(\boldsymbol{\phi}; \boldsymbol{\phi}^{(t)})$.

Bien que cet algorithme augmente explicitement la vraisemblance complète à chaque itération, on peut prouver que cela revient à augmenter la vraisemblance marginale, ce qui est recherché au final. C'est juste qu'augmenter la vraisemblance complète est souvent bien plus facile. L'algorithme EM garantit de trouver un maximum mais celui peut être local et non global, d'où souvent la nécessité d'utiliser l'algorithme pour différentes valeurs initiales. Par rapport à d'autres méthodes, comme celle de Newton, l'algorithme EM ne nécessite pas de connaître les dérivées de la vraisemblance.

Dans le cas du modèle linéaire mixte, la vraisemblance des données complètes peut se décomposer selon chaque composante de la variance: $\mathcal{L}(\boldsymbol{\phi}; \boldsymbol{x}) = p(\boldsymbol{x}|\boldsymbol{\phi}) \propto p(\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{u}, \sigma^2) \times p(\boldsymbol{u}|\sigma_u^2)$;

ce qui donne:

- $l(\sigma^2) = -\frac{n}{2}\log 2\pi - \frac{n}{2}\log\sigma^2 - \frac{\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}}{2\sigma^2}$;
- $l(\sigma_u^2) = -\frac{q}{2}\log 2\pi - \frac{q}{2}\log\sigma_u^2 - \frac{\boldsymbol{u}^T\boldsymbol{u}}{2\sigma_u^2}$.

Pour l'étape E, on obtient $Q(\boldsymbol{\phi}; \boldsymbol{\phi}^{(t)}) = E[\, l(\sigma^2) \mid \boldsymbol{\beta}, \boldsymbol{u}|\boldsymbol{y}, \sigma^{2\,(t)}\,] + E[\, l(\sigma_u^2) \mid \boldsymbol{u}|\sigma_u^{2\,(t)}\,]$:

- ...
- ...

Pour l'étape M, on obtient:

- ...
- ...

Dans les formules ci-dessus, il reste à calculer les espérances.

TODO

4.8 Exemple (suite)

```
sigma.u2.t <- 0.5
sigma2.t <- 1.5
diff1 <- 1
diff2 <- 1
i <- 0
while(diff1 > 10^(-6) & diff2 > 10^(-6)){
  i <- i + 1
  ## ...
  diff1 <- abs(sigma2.hat - sigma2.t)
  diff2 <- abs(sigma.u2.hat - sigma.u2.t)
  sigma2.t <- sigma2.hat
  sigma.u2.t <- sigma.u2.hat
}
```

5 En grande dimension

5.1 Régression d'arête

Dans le contexte de la prédiction génomique, on dispose de données denses de génotypage sur les individus, et elles sont directement utilisées comme prédicteurs. Le jeu de données a donc la particularité que le nombre de variables explicatives est bien plus grand que le nombre d'observations. Dans ce cas, la solution des moindres carrés sur-ajuste le modèle. Il est alors nécessaire d'optimiser le critère des moindres carrés en ajoutant un terme de pénalité. On parle alors de **régularisation**.

Par exemple, prenons le modèle suivant:

$\boldsymbol{y} = M\boldsymbol{\alpha} + \boldsymbol{\epsilon}$ avec $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$

où $\boldsymbol{y}$ contient n phénotypes, M contient $n \times p$ génotypes aux marqueurs (par exemple des marqueurs bi-alléliques codés additivement) et $\boldsymbol{\alpha}$ correspond aux effets additifs des p marqueurs.

Dans le cas où $p >> n$, l'un des modèles possibles est celui de la **régression d'arête** (*ridge regression, RR*) qui régularise via (le carré de) la norme L^2 pour minimiser le critère suivant:

$$||\boldsymbol{y} - M\boldsymbol{\alpha}||_2^2 + \lambda_R ||\boldsymbol{\alpha}||_2^2$$

où le deuxième terme correspond à la pénalité sur la somme des effets des marqueurs pour éviter qu'elle ne soit trop grande, la valeur de λ_R étant généralement choisie par validation croisée.

Sa solution est: $\hat{\boldsymbol{\alpha}}_{RR} = (M^T M + \lambda_R I)^{-1} M^T \boldsymbol{y}$, équivalente à celle du $BLUP$ ci-dessus.

5.2 D'autres pénalités

Un autre modèle est le ***least absolute shrinkage and selection operator, LASSO***, qui régularise via la norme L^1 pour minimiser le critère suivant:

$$||\boldsymbol{y} - M\boldsymbol{\alpha}||_2^2 + \lambda_L ||\boldsymbol{\alpha}||_1$$

Sa solution n'admet pas de forme fermée (*closed-form*), la norme L^1 n'étant pas différentiable en 0, et doit donc être obtenue de manière itérative (plusieurs algorithmes existent).

Il est aussi possible de combiner plusieurs pénalités, comme avec le modèle **Elastic Net**, qui minimise:

$$||\boldsymbol{y} - M\boldsymbol{\alpha}||_2^2 + \lambda_{E1} ||\boldsymbol{\alpha}||_1 + \lambda_{E2} ||\boldsymbol{\alpha}||_2^2$$

D'autres modèles existent, comme le *group LASSO* (pour sélectionner des groupes de variables préalablement définis, par exemple les différents haplotypes d'une même région génomique), le *fused LASSO* (pour pénaliser les différences entre coefficients, par exemple ceux de marqueurs génétiques le long du même chromosome), etc, mais ils ne seront pas détaillés ici.

On utilise le carré de la norme L^2 mais directement la norme L^1, car il est plus facile de manipuler les équations avec des sommes.

5.3 Noyaux

Revenons à la régression d'arête. Il est possible d'exprimer sa solution, $\hat{\boldsymbol{\alpha}}_{RR} = (M^T M + \lambda_R I_p)^{-1} M^T \boldsymbol{y}$, sous une autre forme qui ouvre d'intéressantes perspectives.

Partons de l'égalité suivante:

$(M^T M + \lambda_R I_p) M^T = M^T (M M^T + \lambda_R I_n)$

Multiplier chaque côté à gauche et à droite par les inverses, mène à:

$M^T (M M^T + \lambda_R I_n)^{-1} = (M^T M + \lambda_R I_p)^{-1} M^T$

On obtient donc:

$\hat{\boldsymbol{\alpha}}_{RR} = M^T \hat{\boldsymbol{\delta}}_{RR}$ où $\hat{\boldsymbol{\delta}}_{RR} = (M M^T + \lambda_R I_n)^{-1} \boldsymbol{y}$

L'intérêt immédiat de la deuxième solution, $\hat{\boldsymbol{\delta}}_{RR}$, est de seulement requérir l'inversion d'une matrice $n \times n$ et non $p \times p$. Mais son intérêt principal est ailleurs, plus évident lorsque l'on écrit les valeurs ajustées:

$\hat{\boldsymbol{y}} = M \hat{\boldsymbol{\alpha}}_{RR} = M M^T \hat{\boldsymbol{\delta}}_{RR} = M M^T (M M^T + \lambda_R I_n)^{-1} \boldsymbol{y} = K (K + \lambda_R I_n)^{-1} \boldsymbol{y}$

où $K = M M^T$, c'est-à-dire que, $\forall i, i' \in \{1, \ldots, n\}$, $K_{ii'}$ est le produit scalaire de M_i (la i-ème ligne de M) avec $M_{i'}$ (la i'-ème colonne de M^T): $K_{ii'} = M_i^T M_{i'} = \langle M_i, M_{i'} \rangle$. Cette matrice est notée K pour *kernel* (noyau).

Etant donné deux vecteurs dans $\mathbb{R}^n$, un **noyau** est une fonction qui permet de calculer le produit scalaire entre ces deux vecteurs implicitement dans un espace à plus grande dimension $\mathbb{R}^p$ *sans transformer les vecteurs* ! C'est ce qu'on appelle **l'astuce du noyau** (*kernel trick*):

$\forall M_i, M_{i'} \in \mathbb{R}^n,\ K(M_i, M_{i'}) = \langle \phi(M_i), \phi(M_{i'}) \rangle$ où $\phi : \mathbb{R}^n \to \mathbb{R}^p$

En prédiction génomique, l'intérêt est de pouvoir prendre en compte des interactions non-linéaires (épistatiques) entre colonnes de M (correspondant aux marqueurs génétiques) grâce à la fonction ϕ, simplement via des produits scalaires peu coûteux à calculer (de Los Campos et coll., 2010).

Il existe plusieurs noyaux, comme par exemple le noyau gaussien, $K(M_i, M_{i'}) = \exp^{-\theta\, d(M_i, M_{i'})}$, où d est une distance entre les deux vecteurs, par exemple $||M_i - M_{i'}||_2$, et θ est un paramètre à optimiser, généralement par validation croisée. Le choix du noyau est un sujet en soi et ne sera pas détaillé ici.

6 Références

D'accès gratuit et en français:

— Dempfle (1977): "Relation entre BLUP (Best Linear Unbiased Prediction) et estimateurs bayésiens"
— Foulley (2002): "Méthodes du maximum de vraisemblance en modèle linéaire mixte"
— Foulley (2002): "Algorithme EM : théorie et application au modèle mixte"
— Dagnelie (2012): "Principes d'expérimentation: planification des expériences et analyses de leurs résultats"
— Robert (2001): "L'analyse statistique bayésienne"

7 Annexe

```
t1 <- proc.time(); t1 - t0
```

```
##    user  system elapsed
##   0.773   0.827   0.580
```

```r
print(sessionInfo(), locale=FALSE)
```

```
## R version 3.4.4 (2018-03-15)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.4 LTS
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblasp-r0.2.18.so
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  base
##
## other attached packages:
## [1] MASS_7.3-50   knitr_1.20    rmarkdown_1.9
##
## loaded via a namespace (and not attached):
##  [1] compiler_3.4.4  backports_1.1.2 magrittr_1.5    rprojroot_1.3-2
##  [5] tools_3.4.4     htmltools_0.3.6 yaml_2.1.19     Rcpp_0.12.17
##  [9] stringi_1.2.2   methods_3.4.4   stringr_1.3.1   digest_0.6.15
## [13] evaluate_0.10.1
```