Annexe A : Exemple de mise en oeuvre de modèle mixte sur données réelles et simulées

Benjamin Heuclin, Statisticien, UR AIDA, CIRAD

23/12/2024

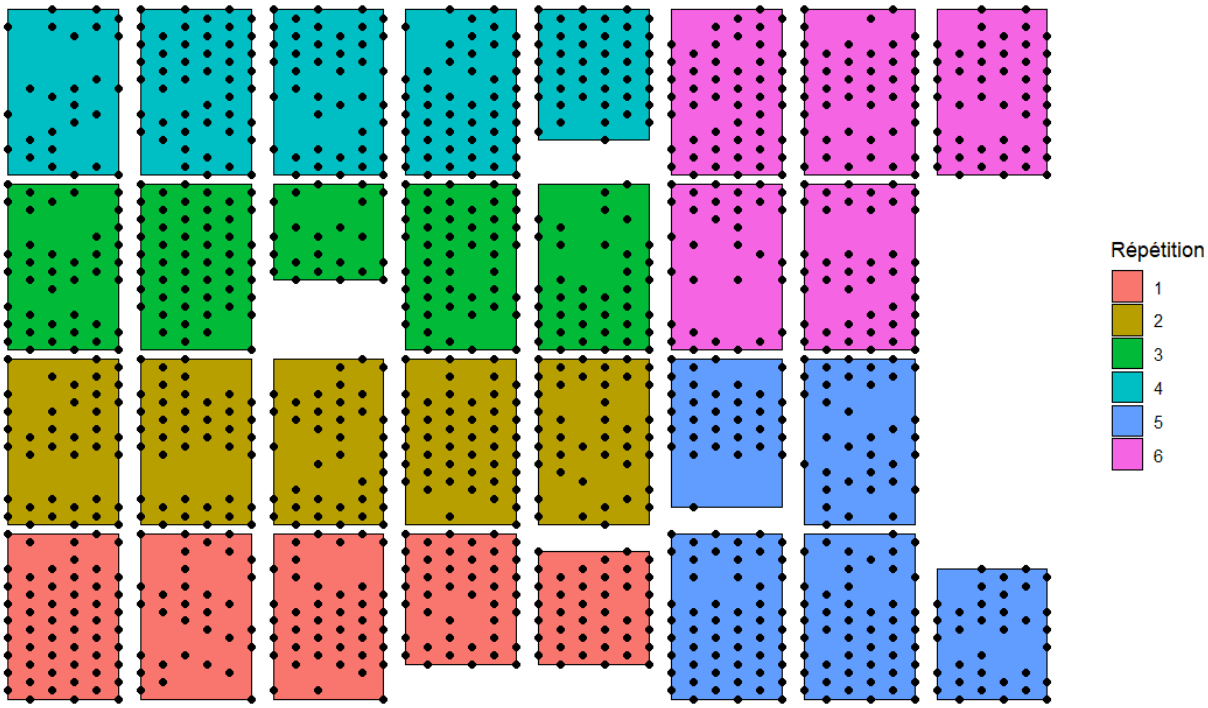
Contents

---

1 Analyse des données de Palmier à Huile issues d'un rassemblement d'essai Alpha-Lattice

Regroupement de 25 essais Alpha-Lattice menés sur le palmier à huile. Les plans Alpha-Lattice sont des plans en blocs incomplets (impossibilité de faire tenir tous les traitements dans chaque bloc) dont les blocs sont rassemblés en répétitions complètes.

La figure suivante présente le plan ALpha-Lattice du 1er essai (ALGP11). Les points représentent les palmiers (les trous sont dus au décès), les rectangles représentent les blocs et les couleurs représentent les répétitions complètes.



Chaque arbre est identifié par un numéro d'essai, de répétition, de bloc, de parcelle et de position dans la parcelle. Les mesures enregistrées incluent le nombre de grappes de fruits sur une période de 3 à 5 ans, le poids des grappes de fruits frais sur une période de 3 à 5 ans (exprimé en kilogrammes), et le poids moyen des fruits sur une période de 3 à 5 ans (exprimé en kilogrammes). Au total, 452 génotypes (traitements) différents sont présents et les données comportent 30 409 observations/individus sur l'ensemble des 25 essais. Pour l'ensemble des individus, nous disposons de leurs pedigrees sur plusieurs générations.

Pour analyser ces données, 2 types de modèles sont possibles :

- Le modèle père/mère qui est formulé en incluant des termes pour les effets fixes des génotypes du père et de la mère, ainsi qu'un terme pour l'erreur aléatoire.
- Et le modèle animal qui prend en compte les contributions génétiques non seulement des parents mais aussi des ancêtres plus éloignés.

Nous allons ici utiliser un modèle animal.

Ce jeu de données généralise les exemples n°1 « Essai agronomique en blocs complets randomisés » et n°2 « Essai agronomique génétique ».

Importation des données

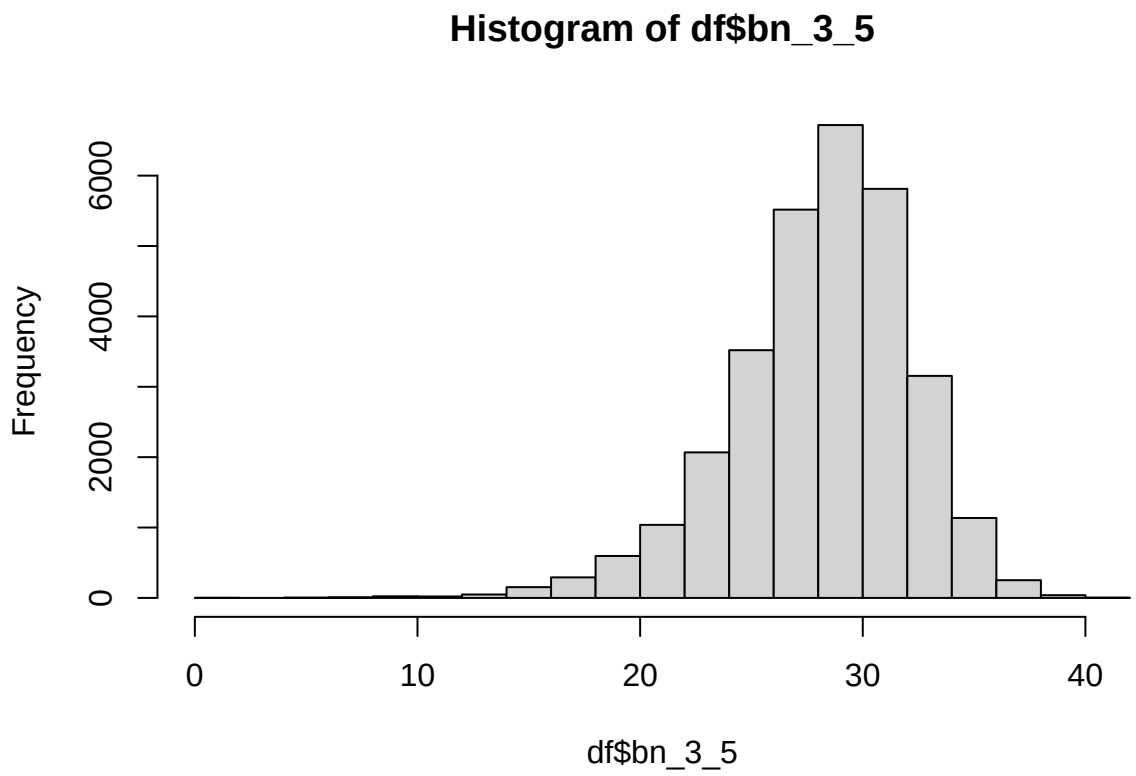
```
### Donnees de production moyenne 3-5 ans
df <- read.csv2("Data/Données_palmier_multi_essais/prod_3_5.csv")
### Donnees des pédigrés
ped <- read.csv2("Data/Données_palmier_multi_essais/ped_3_5.csv")
ped <- nadiv::prepPed(ped)
```

```
df$treef_id <- factor(df$treef_id)
df$trial <- factor(df$trial)
df$replicate <- factor(paste0(df$trial, "_R", df$replicate))
```

```r
df$block <- factor(paste0(df$replicate, df$block))
df$plot <- factor(paste0(df$block, "_P", df$plot))

### Creation de la variable croisement
df$cross <- factor(paste(df$mother_id, df$father_id, sep = " x "))
df$mother_id <- factor(df$mother_id)
df$father_id <- factor(df$father_id)
```

Ici, nous souhaitons analyser le nombre de régimes produits par les palmiers entre 3 et 5 ans

```r
hist(df$bn_3_5)
```



1.1 Un seul essai (ALGP11)

Nous commençons dans un premier temps par analyser d'un seul essai pour pouvoir mettre en oeuvre un maximum de librairies.

On utilise un modèle *animal* pour estimer l'effet des génotypes :

$$y = \mu + u + Z_r r + Z_b b + Z_p p + \varepsilon$$

avec :

- y le nombre de régimes produits entre 3 et 5 ans

- μ l'intercept
- u l'effet aléatoire génotype, $u \sim N(0, \sigma_u^2 A)$ avec A la matrice de parenté entre les génotypes
- r l'effet aléatoire (iid) des répétitions
- b l'effet aléatoire (iid) des blocs
- p l'effet aléatoire (iid) des parcelles
- ε les résidus supposés iid
- Z_r, Z_b et Z_p représentent les matrice de design associées à chacun des effets aléatoires r, b et p respectivement.

Préparation des données

```
df_1 = df %>% filter(trial == "ALGPO1")

df_1$treef_id = as.factor(as.character(df_1$treef_id))

# Préparation matrice d'apparentement
descendants_names = as.character(df_1$treef_id)

ped1 <- prunePed(ped, phenotyped = descendants_names)

genitor_names = unique(as.character(ped1$treef_id[!ped1$treef_id %in% descendants_names]))

# Calcul de la matrice de Relationship (Wright, 1922) qui est le double de la
# matrice de coefficients de parenté
A <- 2*kinship2::kinship(id = ped1$treef_id, momid = ped1$mother_id, dadid = ped1$father_id)

A11 = A[descendants_names, descendants_names]
A21=A[genitor_names, descendants_names]
```

ATTENTION : brms a tourné avec R 4.2.2 et non avec R 4.4.1 et INLA a tourné avec R 4.4.1 mais pas avec R 4.2.2

1.1.1 lme4GS

Avec le package lme4gs, nous ajoutons une matrice de parenté en utilisant la fonction `Uvcov`. Cette matrice a été créée plus tôt dans le document. Il a également été ajouté les croisements.

```
library(lme4GS)
fit_lme4gs = lmerUvcov(bn_3_5~ 1  + (1|treef_id) + (1|replicate) +(1|block) + (1|plot),
                       data = df_1,
                       Uvcov=list(treef_id=list(K=A11)),
                       verbose=FALSE)
summary(fit_lme4gs)
```

Calcul du R2

```
cor(df_1$bn_3_5, predict(fit_lme4gs))^2
```

```
## [1] 0.5113257
```

1.1.2 sommer

Avec le package sommer, on vient intégrer une matrice génétique dans les effets aléatoires par l'argument vsr.

```
library(sommer)
fit_sommer <- mmer(bn_3_5 ~ 1,
                   random =  ~ vsr(treef_id, Gu=A11) + replicate + block + plot,
                   data=df_1, verbose = FALSE, dateWarning = FALSE, nIters = 3)
summary(fit_sommer)
```

Calcul du R2

```
fitted_sommer = fitted(fit_sommer)
```

```
## Version out of date. Please update sommer to the newest version using:
## install.packages('sommer') in a new session
##  Use the 'dateWarning' argument to disable the warning message.Version out of date. Please update so
## install.packages('sommer') in a new session
##  Use the 'dateWarning' argument to disable the warning message.
```

```
cor(fitted_sommer$dataWithFitted$bn_3_5.fitted, df_1$bn_3_5)^2
```

```
## [1] 0.5095342
```

1.1.3 BGLR

Avec le package BGLR, il faut specifier les matrice de design des effets aléatoires.

```
library(BGLR)

Z_rep = design.matrix(df_1$replicate)
Z_block <- design.matrix(df_1$block)
Z_plot <- design.matrix(df_1$plot)


ETA = list(
  u = list(K = A11, model = "RKHS"),
  rep = list(K=tcrossprod(Z_rep), model = "RKHS", data = df_1),
  block = list(K=tcrossprod(Z_block), model = "RKHS", data = df_1),
  plot = list(K=tcrossprod(Z_plot), model = "RKHS", data = df_1)
)

dir.create("fit_BGLR/")
```

```
## Warning in dir.create("fit_BGLR/"): 'fit_BGLR' existe déjà
```

```
fit_BGLR = BGLR(y = df_1$bn_3_5, ETA = ETA,
                nIter =5000, burnIn =2000, saveAt ="fit_BGLR/", verbose = FALSE)
```

Ensuite, on vient estimer les variances associées à chaque effet aléatoire, ainsi que la variance résiduelle.

```r
# Variance résiduelle
fit_BGLR$varE

# Effets aléatoires
fit_BGLR$ETA$u$varU
fit_BGLR$ETA$rep$varU
fit_BGLR$ETA$block$varU
fit_BGLR$ETA$plot$varU
```

Calcul du R2

```r
cor(fit_BGLR$yHat, df_1$bn_3_5)^2
```

```
## [1] 0.4965567
```

1.1.4 BGGE

Avec le package BGGE, il faut specifier les matrice de design des effets aléatoires.

```r
library(BGGE)

Z_rep <- contrasts(as.factor(df_1$replicate), contrasts = FALSE)[df_1$replicate, ]
Z_block <- contrasts(as.factor(df_1$block), contrasts = FALSE)[df_1$block, ]
Z_plot <- contrasts(as.factor(df_1$plot), contrasts = FALSE)[df_1$plot, ]

K = list(
  u = list(Kernel = A11, Type = "D"),
  rep = list(Kernel=tcrossprod(Z_rep), Type = "D"),
  block = list(Kernel=tcrossprod(Z_block), Type = "D"),
  plot = list(Kernel=tcrossprod(Z_plot), Type = "D")
)

fit_BGGE = BGGE(y = df_1$bn_3_5,
                K = K,
                XF = NULL,
                ne=as.vector(c(nrow(df_1))),
                ite = 5000, burn = 2000, thin = 3)
```

```r
fit_BGGE$varE
fit_BGGE$K$u$varu
fit_BGGE$K$rep$varu
fit_BGGE$K$block$varu
fit_BGGE$K$plot$varu
```

Calcul du R2

```r
cor(fit_BGGE$yHat, df_1$bn_3_5)^2
```

```
## [1] 0.6226991
```

1.1.5 BRMS

```
library(brms)
priors <- c(prior(normal(0, 100), class = "Intercept"),
            prior(normal(0, 100), class = "b")
)

fit_brms <- brms::brm(
  bf(bn_3_5 ~ 1 + (1|gr(treef_id, cov=A11)) + (1|replicate) + (1|block) + (1|plot)),
  family = gaussian(),
  data = df_1,
  data2 = list(A11=A11),
  iter=5000, thin = 5, cores=4, chains=4)

summary(fit_brms)
```

1.1.6 INLA

Calcul de l'inverse de la matrice d'apparentement. Utilisation du package "pedigreemm"

```
names(ped1) = c("Individual", "Parent1", "Parent2")

ped1$Individual = as.character(ped1$Individual)
ped1$Parent1 = as.character(ped1$Parent1)
ped1$Parent2 = as.character(ped1$Parent2)

library(pedigreemm)
ped1_inla = pedigreemm::pedigree(sire = ped1$Parent2, dam = ped1$Parent1, label = ped1$Individual)

T_Inv <- as(ped1_inla, "sparseMatrix")
```

```
## 'as(<dtTMatrix>, "dtCMatrix")' est obsolète.
## Utilisez plutôt 'as(., "CsparseMatrix")'.
## Voir help("Deprecated") et help("Matrix-deprecated").
```

```
# returns a sparse, unit lower-triangular matrix which is the inverse of the "L"
# part of the "LDL'" form of the Cholesky factorization of the relationship matrix.
# All non-zero elements below the diagonal are -0.5.
D_Inv <- Matrix::Diagonal(x=1/pedigreemm::Dmat(ped1_inla))
Ainv_pedigreemm <- t(T_Inv) %*% D_Inv %*% T_Inv
dimnames(Ainv_pedigreemm)[[1]] <- dimnames(Ainv_pedigreemm)[[2]] <- ped1_inla@label

map = data.frame(Individual = ped1_inla@label, newId = 1:length(ped1_inla@label))
map$newId = as.numeric(map$newId)
```

```
df1_inla = df_1
df1_inla$Individual = as.character(df1_inla$treef_id)
df1_inla = left_join(df1_inla, map)
```

```
## Joining with `by = join_by(Individual)`
```

```r
library(INLA)
```

```
## Le chargement a nécessité le package : sp
```

```
## This is INLA_24.05.10 built 2024-05-10 19:59:04 UTC.
##  - See www.r-inla.org/contact-us for how to get help.
##  - List available models/likelihoods/etc with inla.list.models()
##  - Use inla.doc(<NAME>) to access documentation
```

```r
##lois a priori
#Prior sur la variance des effets al?atoires (effets blocs et environnement) :
#Loi "Penalized complexity" param?tr?e de fa?on qu'il y ait une proba a priori inf?rieure ? 1% pour
#que ces variances soient sup?rieures ? un seuil fix? ? 10 fois la variance totale de la variable ?tudi
vtot <- var(df1_inla$bn_3_5, na.rm=TRUE)
s2max <- 10*vtot
prec.Random <- list(prior="pc.prec", param=c(s2max,0.01), fixed=FALSE)

#M?me prior pour la variance des effets g?n?tique
prec.A <- list(prior="pc.prec", param=c(s2max, 0.01), fixed=FALSE)

#Appel INLA en utilisant la structure "generic0" qui permet
#de formuler le modele avec "sigma2g" (variance g?n?tique)
#et "sigma2e" (variance environnementale) comme parametres
#et de calculer un DIC

fit_inla <- inla(bn_3_5 ~ 1 +
                   f(replicate, model="iid", constr=FALSE, hyper=list(theta=prec.Random)) +
                   f(block, model="iid", constr=FALSE, hyper=list(theta=prec.Random)) +
                   f(plot, model="iid", constr=FALSE, hyper=list(theta=prec.Random)) +
                   f(newId, model="generic0", Cmatrix=Ainv_pedigreemm, constr=FALSE,
                     hyper=list(theta=prec.A)),
                 data=df1_inla,
                 family="gaussian",
                 control.family=list(hyper=list(theta=prec.Random)),
                 control.compute=list(dic=TRUE))
```

```r
summary(fit_inla)
```

```r
# les variances résiduelle et génétique
var_est = 1/fit_inla$summary.hyperpar$mean; var_est
```

```
## [1] 7.75213086 0.01477379 0.38242038 0.48535035 3.27418763
```

```r
names(var_est) = c("residuals", summary(fit_inla)$random.names)
print(var_est)
```

```
##  residuals  replicate      block       plot      newId
## 7.75213086 0.01477379 0.38242038 0.48535035 3.27418763
```

```
# var_est['newId'] / sum(var_est) # H2
```

Calcul du R2

```
fitted_inla = fit_inla$summary.fitted.values$mean
cor(fitted_inla, df1_inla$bn_3_5)^2
```

```
## [1] 0.523558
```

```
fitted_inla = fit_inla$summary.fitted.values$mean
plot(fitted_inla, df1_inla$bn_3_5); abline(0, 1)
```

1.1.7 AsReml

```
library(asreml)
```

```
## Loading ASReml-R version 4.2
```

```
##
## Attachement du package : 'asreml'
```

```
## L'objet suivant est masqué depuis 'package:sommer':
##
##     vpredict
```

```
## Les objets suivants sont masqués depuis 'package:MASS':
##
##     coop, oats
```

```
ai <- asreml::ainverse(ped1)
fit_asreml <- asreml(bn_3_5 ~ 1,
                     random = ~vm(treef_id, ai)  + replicate + block + plot,
                     data = df_1)
detach("package:asreml", unload = TRUE)
summary(fit_asreml)$varcomp
```

Calcul du R2

```
fitted_asreml = fit_asreml$linear.predictors
cor(df_1$bn_3_5, fitted_asreml)^2
```

```
## [1] 0.5112128
```

Résumé des estimations des composantes de la variance :

Librairie	Parenté	Répétition	Block	Parcelle	Résidus
lme4GS	3.27	0	0.46	0.52	7.84
sommer	3.22	0.0034	0.56	0.51	7.85
BGLR	3.53	0.02	0.21	0.51	7.83
BGGE	5.76	16.22	5.10	2.34	7.03
brms	3.42	0.11	0.45	0.53	7.78
INLA	3.27	0.02	0.38	0.49	7.75
AsReml	3.28	0	0.46	0.52	7.84

1.2 Tous les essais

On souhaite maintenant analyser l'ensemble des 25 essais comptabilisant 30409 observations. Seul INLA et AsReml ont pu ajuster le modèle statistique. Les autres librairies plantent.

On utilise le même modèle *animal* que précedemment, mais en ajoutant un effet aléatoire Essai :

$$y = \mu + u + Z_t t + Z_r r + Z_b b + Z_p p + \varepsilon$$

avec :

- y le nombre de régimes produits entre 3 et 5 ans
- μ l'intercept
- u l'effet aléatoire génotype, $u \sim N(0, \sigma_u^2 A)$ avec A la matrice de parenté entre les génotypes
- t l'effet aléatoire (iid) de l'essai (représentant différentenvironnement)
- r l'effet aléatoire (iid) des répétitions
- b l'effet aléatoire (iid) des blocs
- p l'effet aléatoire (iid) des parcelles
- ε les résidus supposés iid
- Z_t, Z_r, Z_b et Z_p représentent les matrice de design associées à chacun des effets aléatoires t, r, b et p respectivement.

Préparation des données

```
# Préparation matrice d'apparentement
descendants_names = as.character(df$treef_id)

ped_all <- prunePed(ped, phenotyped = descendants_names)
```

```
## Warning in numPed(pedigree[, 1:3]): Dams appearing as Sires - assumed selfing
## in pedigree
```

```
genitor_names = unique(as.character(ped_all$treef_id[!ped_all$treef_id %in% descendants_names]))
```

1.2.1 INLA

Calcul de l'inverse de la matrice d'apparentement. Utilisation du package "pedigreemm"

```r
names(ped_all) = c("Individual", "Parent1", "Parent2")

ped_all$Individual = as.character(ped_all$Individual)
ped_all$Parent1 = as.character(ped_all$Parent1)
ped_all$Parent2 = as.character(ped_all$Parent2)

library(pedigreemm)
ped_all_inla = pedigreemm::pedigree(sire = ped_all$Parent2, dam = ped_all$Parent1, label = ped_all$Indi

T_Inv <- as(ped_all_inla, "sparseMatrix")
# returns a sparse, unit lower-triangular matrix which is the inverse of the "L"
# part of the "LDL'" form of the Cholesky factorization of the relationship matrix.
# All non-zero elements below the diagonal are -0.5.
D_Inv <- Matrix::Diagonal(x=1/pedigreemm::Dmat(ped_all_inla))
Ainv_pedigreemm <- t(T_Inv) %*% D_Inv %*% T_Inv
dimnames(Ainv_pedigreemm)[[1]] <- dimnames(Ainv_pedigreemm)[[2]] <- ped_all_inla@label

map = data.frame(Individual = ped_all_inla@label, newId = 1:length(ped_all_inla@label))
map$newId = as.numeric(map$newId)
```

```r
df_inla = df
df_inla$Individual = as.character(df_inla$treef_id)
df_inla = left_join(df_inla, map)
```

```
## Joining with `by = join_by(Individual)`
```

```r
library(INLA)
##lois a priori
#Prior sur la variance des effets al?atoires (effets blocs et environnement) :
#Loi "Penalized complexity" param?tr?e de fa?on qu'il y ait une proba a priori inf?rieure ? 1% pour
#que ces variances soient sup?rieures ? un seuil fix? ? 10 fois la variance totale de la variable ?tudi
vtot <- var(df_inla$bn_3_5, na.rm=TRUE)
s2max <- 10*vtot
prec.Random <- list(prior="pc.prec", param=c(s2max,0.01), fixed=FALSE)

#M?me prior pour la variance des effets g?n?tique
prec.A <- list(prior="pc.prec", param=c(s2max, 0.01), fixed=FALSE)

#Appel INLA en utilisant la structure "generic0" qui permet
#de formuler le modele avec "sigma2g" (variance g?n?tique)
#et "sigma2e" (variance environnementale) comme parametres
#et de calculer un DIC

fit_inla <- inla(bn_3_5 ~ 1 +
                   f(trial, model="iid", constr=FALSE, hyper=list(theta=prec.Random)) +
                   f(replicate, model="iid", constr=FALSE, hyper=list(theta=prec.Random)) +
                   f(block, model="iid", constr=FALSE, hyper=list(theta=prec.Random)) +
                   f(plot, model="iid", constr=FALSE, hyper=list(theta=prec.Random)) +
                   f(newId, model="generic0", Cmatrix=Ainv_pedigreemm, constr=FALSE,
                     hyper=list(theta=prec.A)),
                 data=df_inla,
                 family="gaussian",
```

```
                control.family=list(hyper=list(theta=prec.Random)),
                control.compute=list(dic=TRUE))
```

```
summary(fit_inla)
```

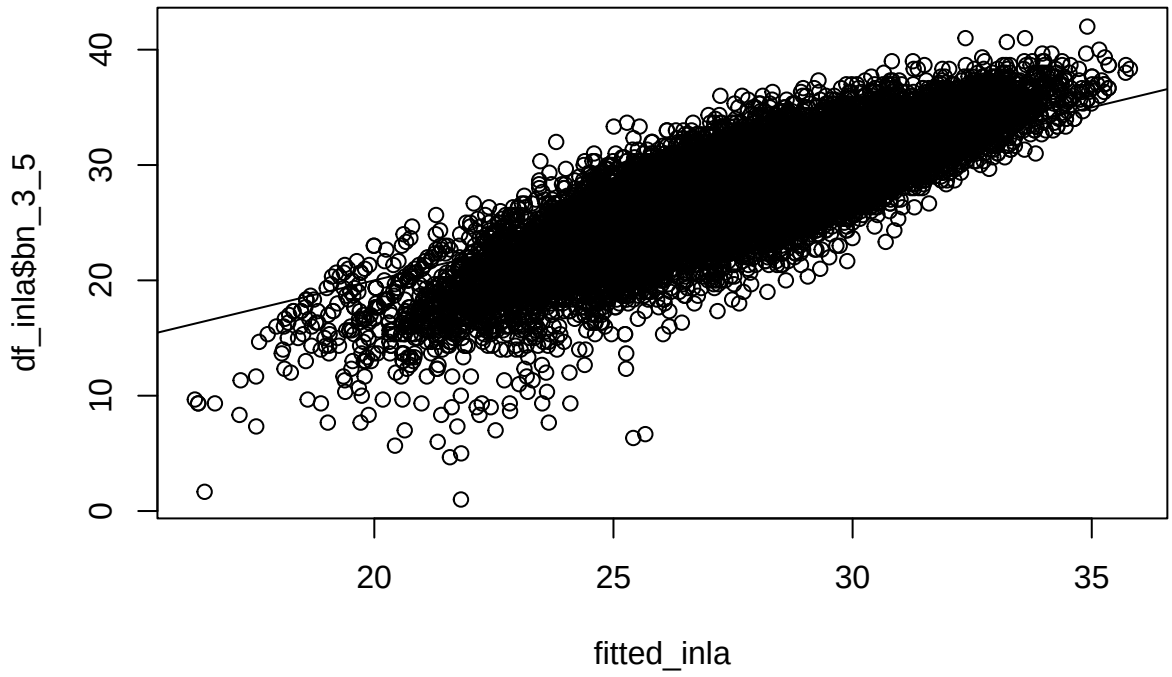
```
## Time used:
##     Pre = 2.8, Running = 116, Post = 5.3, Total = 124
## Fixed effects:
##              mean    sd 0.025quant 0.5quant 0.975quant  mode kld
## (Intercept) 28.35 0.594     27.185    28.35     29.516 28.35   0
##
## Random effects:
##   Name     Model
##     trial IID model
##    replicate IID model
##    block IID model
##    plot IID model
##    newId Generic0 model
##
## Model hyperparameters:
##                                          mean    sd 0.025quant 0.5quant
## Precision for the Gaussian observations 0.135 0.004      0.128    0.135
## Precision for trial                     1.118 0.482      0.540    1.007
## Precision for replicate                 3.363 0.617      2.253    3.326
## Precision for block                     5.414 1.139      3.517    5.298
## Precision for plot                      1.479 0.117      1.260    1.475
## Precision for newId                     0.139 0.010      0.121    0.138
##                                         0.975quant  mode
## Precision for the Gaussian observations      0.142 0.135
## Precision for trial                          2.378 0.806
## Precision for replicate                      4.671 3.290
## Precision for block                          7.982 5.073
## Precision for plot                           1.719 1.470
## Precision for newId                          0.160 0.137
##
## Deviance Information Criterion (DIC) ...............: 155899.30
## Deviance Information Criterion (DIC, saturated) ....: 39172.34
## Effective number of parameters .....................: 8797.02
##
## Marginal log-Likelihood:  -96226.15
##  is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```
# les variances résiduelle et génétique
var_est = 1/fit_inla$summary.hyperpar$mean
names(var_est) = c("residuals", summary(fit_inla)$random.names)
print(var_est)
```

```
## residuals     trial replicate     block      plot     newId
## 7.4021030 0.8943494 0.2973964 0.1847088 0.6761076 7.1968729
```

```
# var_est['newId'] / sum(var_est) # H2
```

```
fitted_inla = fit_inla$summary.fitted.values$mean
plot(fitted_inla, df_inla$bn_3_5); abline(0, 1)
```




```
cor(fitted_inla, df_inla$bn_3_5)^2
```

```
## [1] 0.718254
```

1.2.2 AsReml

```
library(asreml)
```

```
## Loading ASReml-R version 4.2

##
## Attachement du package : 'asreml'

## L'objet suivant est masqué depuis 'package:sommer':
##
##     vpredict
```

```
## Les objets suivants sont masqués depuis 'package:MASS':
##
##     coop, oats
```

```
ai <- asreml::ainverse(ped_all)
fit_asreml <- asreml(bn_3_5 ~ 1,
                     random = ~vm(treef_id, ai) + trial + replicate + block + plot,
                     data = df)
```

```
## ASReml Version 4.2 30/01/2025 16:56:57
##           LogLik        Sigma2     DF     wall
##  1     -51808.47      9.631542  30408   16:56:58
##  2     -51659.20      9.504717  30408   16:56:58
##  3     -51554.47      9.174438  30408   16:56:58
##  4     -51489.97      8.620462  30408   16:56:59
##  5     -51461.20      7.880454  30408   16:56:59
##  6     -51457.60      7.477685  30408   16:56:59
##  7     -51457.55      7.420263  30408   16:56:59
##  8     -51457.55      7.420691  30408   16:57:01
```

```
detach("package:asreml", unload = TRUE)
```

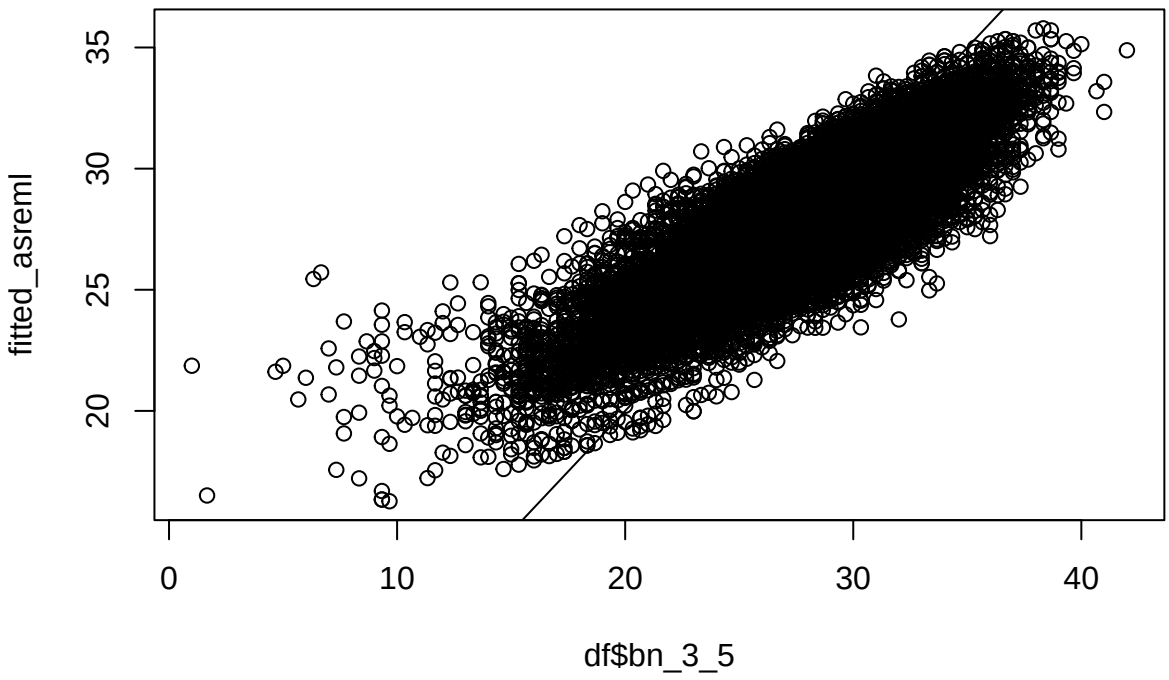
```
## License checkin Thu Jan 30 16:57:01 2025
```

```
summary(fit_asreml)$varcomp
```

```
## Online License checked out Thu Jan 30 16:57:02 2025
```

```
##                  component  std.error   z.ratio bound %ch
## trial            0.9383043 0.30130995  3.114083     P   0
## replicate        0.2968917 0.05489120  5.408731     P   0
## block            0.1916281 0.03939289  4.864536     P   0
## plot             0.6766286 0.05295447 12.777554     P   0
## vm(treef_id, ai) 7.1843841 0.52073093 13.796730     P   0
## units!R          7.4206908 0.19375352 38.299643     P   0
```

```
fitted_asreml = fit_asreml$linear.predictors
plot(df$bn_3_5 , fitted_asreml); abline(0, 1)
```




```
cor(df$bn_3_5, fitted_asreml)^2
```

```
## [1] 0.715922
```

Résumé des estimations des composantes de la variance et temps d'exécution

Librairie	Parenté	Essai	Répétition	Block	Parcelle	Résidus	Temps d'exécution
INLA	7.21	0.96	0.30	0.19	0.68	7.40	60s
AsReml	7.19	0.94	0.30	0.19	0.67	7.41	5s

---

2 Analyse de données longitudinales : cas d'une étude clinique

L'étude EVA (Epidémiologie du Vieillissement Artériel) avait pour objectif principal de mieux comprendre les processus du vieillissement normal et les conséquences du processus de vieillissement sur les fonctions cognitives et les pathologies cardio-vasculaires. Entre juin 1991 et juin 1993, 1389 sujets volontaires (574 hommes et 815 femmes), âgés de 59 à 71 ans, ont été recrutés pour participer à cette étude. Le but de l'analyse était d'étudier l'évolution du sélénium dans le temps et de rechercher les facteurs associés à cette évolution. Dans EVA, le sélénium plasmatique a été mesuré à l'inclusion, 2 ans, 6 ans et 9 ans. La variable V1 correspond aux sujets de l'étude. La variable levelstudy qui correspond aux niveaux d'études des participants avec 0 = aucun niveau, 1 = Certif étude / CAP et 2 = Brevet / Bac. Et la variable sex avec 1 = Homme, 2 = Femme

L'analyse d'un tel dispositif est alors possible à l'aide d'un modèle à pentes et intercepts aléatoires :

$$Y_{tijk} = \mu + X_{tijk}\beta + E_i + S_j + u_k + X_{tijk}b_k + \varepsilon_{tijk}$$

Où

- t est l'indice des temps de mesure (à l'inclusion, 2 ans, 6 ans et 9 ans)
- i est l'indice du niveaux d'étude (0, 1, 2)
- j est l'indice pour le sexe (1 ou 2)
- k est l'indice du sujet
- Y_{tijk} est l'observation du niveau de sélénium plasmatique de l'individus $tijk$
- μ est l'intercept
- X_{tijk} est l'année de messure (0, 2, 6, 9))
- β est l'effet du temps sur le sélénium (pente fixe)
- E_i est l'effet du $i^{\text{ème}}$ niveau d'étude sur le sélénium
- S_j est l'effet du $j^{\text{ème}}$ sexe sur le sélénium
- u_k est l'intercepte aléatoire du $k^{\text{ème}}$ sujet de l'étude,
- b_k est la pente aléatoire du $k^{\text{ème}}$ sujet de l'étude
- ε_{tijk} le résidu associé à l'observation $tijk$, tel que le vecteur de l'ensemble des résidus $\varepsilon \sim N_{1389\times4}(0, \sigma_E^2\mathbf{I}_{1389\times4})$

Dans ce modèle à pentes et intercepte aléatoires, on peut faire l'hypothèse que pour chaque individu, son intercept et sa pende aléatoire ne sont pas indépendants. Cela ce traduit alors par la loi jointe suivante :

$$(u_k, b_k)^T \sim N_2(0, G), \forall k$$

où G est une matrice de taille 2×2 de covariance non-structurée inconnue.

Importation des données

```
library(haven)

data <- read_sas(data_file = "Data/Données_clinique_longitudinales/seleniumtp.sas7bdat")
anyNA(data)
```

```
## [1] TRUE
```

```
data2 <- na.omit(data)
dim(data2)
```

```
## [1] 3421   13
```

```r
data2$V1 <- as.factor(data2$V1)
```

Les données consistent en :

- V1 qui correspond aux sujets de l'étude
- levelstudy qui correspond aux niveaux d'études des participants avec :
  - 0 = aucun niveau
  - 1 = Certif étude / CAP
  - 2 = Brevet / Bac
- sex avec :
  - 1 = Homme
  - 2 = Femme

2.1 nlme

```r
library(nlme)
```

```
##
## Attachement du package : 'nlme'

## L'objet suivant est masqué depuis 'package:pedigreemm':
##
##     ranef

## L'objet suivant est masqué depuis 'package:lme4GS':
##
##     ranef

## L'objet suivant est masqué depuis 'package:lme4':
##
##     lmList

## L'objet suivant est masqué depuis 'package:dplyr':
##
##     collapse
```

```r
mod_nlme <- lme(selenium ~ delai,
            random = ~ 1 + delai|V1,
            data = data2
)
summary(mod_nlme)
```

2.2 lme4

```r
library(lme4)
mod_lmer <- lmer(selenium ~ delai + (1 + delai|V1),
            data = data2)
summary(mod_lmer)
```

2.3 Sommer

```r
mod_sommer <- mmer(selenium ~ delai,
                   random = ~ V1 + V1:delai,
                   rcov = ~ units,
                   data = data2)

summary(mod_sommer)
```

2.4 BRMS

```r
library(brms)
```

```
## Le chargement a nécessité le package : Rcpp

## Loading 'brms' package (version 2.21.0). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').

##
## Attachement du package : 'brms'

## L'objet suivant est masqué depuis 'package:pedigreemm':
##
##     ranef

## L'objet suivant est masqué depuis 'package:lme4GS':
##
##     ranef

## L'objet suivant est masqué depuis 'package:lme4':
##
##     ngrps

## L'objet suivant est masqué depuis 'package:stats':
##
##     ar
```

```r
fit_brms <- brms::brm(
  brms::bf(selenium ~ delai + (1 + delai|V1)),
  family = gaussian(),
  data = data2,
  iter = 5000, thin = 5, cores = 4, chains = 4
)
```

```
## Compiling Stan program...

## Start sampling
```

```
summary(fit_brms)
```

---

3 Analyse de données temporelles : Les oiseaux d'Hawaï

Données d'abondance de trois espèces d'oiseaux, mesurées sur trois îles d'Hawaï de 1956 à 2003. On a donc plusieurs séries temporelles.

Données publiées dans *Time series analysis of Hawaiian waterbirds. Analysing ecological data, Reed, Elphick, Zuur, Ieno, Smith (2007), Springer.*

Importation des données

```
hawai <- read.table(file = "Data/Données_temorelles_Hawaii/Hawaii.txt", header = T)
hawai$Birds<-sqrt(hawai$Moorhen.Kauai)
head(hawai)
```

```
##   Year Stilt.Oahu Stilt.Maui Coot.Oahu Coot.Maui Moorhen.Kauai Rainfall
## 1 1956        163        169       528       177             2    15.16
## 2 1957        272        190       338       273            NA    15.48
## 3 1958        549        159       449       256             2    16.26
## 4 1959        533        211       822       170            10    21.25
## 5 1960         NA        232        NA       188             4    10.94
## 6 1961        134        155       717       149            10    19.93
##      Birds
## 1 1.414214
## 2       NA
## 3 1.414214
## 4 3.162278
## 5 2.000000
## 6 3.162278
```

Pour chaque année, on a une note d'abondance moyenne sur les 3 îles ainsi que des variables explicatives dont la pluviométrie (voir l'article pour plus de détails). On peut alors analyser les données à l'aide d'un modèle linéaire avec une structuration AR(1) sur les résidus pour prendre en compte la structuration temporelle des données :

$$y = X\beta + \varepsilon, \qquad \varepsilon \sim N(0, R)$$

avec R de la forme :

3.1 nlme

$$\sigma^2 \begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 \\ \rho & 1 & \rho & \rho^2 \\ \rho^2 & \rho & 1 & \rho \\ \rho^3 & \rho^2 & \rho & 1 \end{bmatrix}$$

Figure 1: Structure AR1

```
# modèle sans correlation temporelle
mod_nlme <- gls(Birds ~ Stilt.Oahu + Stilt.Maui + Coot.Oahu + Coot.Maui + Moorhen.Kauai + Rainfall,
                    # correlation = corAR1(form =~ Year),
                    na.action = na.omit,
                    data = hawai,
                method = "REML")

# modèle avec correlation temporelle
mod_nlme_AR1 <- gls(Birds ~ Stilt.Oahu + Stilt.Maui + Coot.Oahu + Coot.Maui + Moorhen.Kauai + Rainfall,
                    correlation = corAR1(form =~ Year),
                    na.action = na.omit,
                    data = hawai,
                method = "REML")

AIC(mod_nlme, mod_nlme_AR1)
```

```
##              df      AIC
## mod_nlme      8 210.6715
## mod_nlme_AR1  9 190.0797
```

```
summary(mod_nlme_AR1)
```

```
## Generalized least squares fit by REML
##   Model: Birds ~ Stilt.Oahu + Stilt.Maui + Coot.Oahu + Coot.Maui + Moorhen.Kauai +      Rainfall
##   Data: hawai
##        AIC      BIC    logLik
##   190.0797 204.3313 -86.03983
##
## Correlation Structure: ARMA(1,0)
##  Formula: ~Year
##  Parameter estimate(s):
##      Phi1
## 0.8258943
##
## Coefficients:
##                   Value Std.Error   t-value p-value
## (Intercept)    3.856046 1.0512902  3.667917  0.0008
## Stilt.Oahu    -0.000170 0.0007346 -0.231954  0.8179
## Stilt.Maui     0.001156 0.0019652  0.588255  0.5600
## Coot.Oahu      0.000339 0.0006593  0.514082  0.6103
## Coot.Maui      0.000332 0.0019743  0.168197  0.8674
## Moorhen.Kauai  0.047197 0.0038429 12.281496  0.0000
## Rainfall      -0.024764 0.0180392 -1.372804  0.1783
##
```

```
##  Correlation:
##               (Intr) Stlt.O Stlt.M Cot.Oh Coot.M Mrhn.K
## Stilt.Oahu    -0.179
## Stilt.Maui    -0.469 -0.028
## Coot.Oahu     -0.317 -0.183  0.120
## Coot.Maui     -0.053 -0.019 -0.414 -0.167
## Moorhen.Kauai -0.213 -0.074 -0.121  0.216  0.046
## Rainfall      -0.500  0.015  0.365  0.075  0.090 -0.089
##
## Standardized residuals:
##        Min         Q1        Med         Q3        Max
## -1.7017157 -0.5510953  0.5183217  0.7481557  1.2571538
##
## Residual standard error: 1.61041
## Degrees of freedom: 43 total; 36 residual
```

3.2 BRMS

```
library(brms)
fit_brms <- brms::brm(
  bf(Birds ~ Rainfall + Year) + cor_ar(~ Year, p = 1),
  family = gaussian(),
  data = hawai,
  iter = 5000, thin = 5, cores = 4, chains = 4
)
summary(fit_brms)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: Birds ~ Rainfall + Year
##          autocor ~ arma(time = Year, gr = NA, p = 1, q = 0, cov = FALSE)
##    Data: hawai (Number of observations: 45)
##   Draws: 4 chains, each with iter = 5000; warmup = 2500; thin = 5;
##          total post-warmup draws = 2000
##
## Correlation Structures:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## ar[1]     0.78      0.12     0.56     1.02 1.00     1901     1974
##
## Regression Coefficients:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept  -440.64    151.82  -723.68  -141.30 1.00     1853     1681
## Rainfall     -0.01      0.03    -0.07     0.06 1.00     2085     2032
## Year          0.23      0.08     0.08     0.37 1.00     1848     1681
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     1.93      0.22     1.56     2.40 1.00     1903     1974
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Les deux librairies estiment un coefficient d'autocorrélation compris entre 0.7 et 0.8.

---

4 Analyse de données spatiales : Essai agronomique sur le palmier à huile

Dans un essai sur le palmier à huile plantée au Brésil en 2010, on souhaite étudier l'effet de 8 croisements d'hybrides plantés selon des densités variant de 103 arbres/ha à 143 arbres/ha sur le diamètre des troncs à 12 ans. Il y a 2 parcelles élémentaires de 64 arbres par croisement, dont 49 arbres utiles (on ne tient pas compte des palmiers en bordure de parcelle). Les parcelles sont regroupées en blocs incomplets qui sont eu même regroupés en répétitions complètes (plan Alpha-Lattice). Dans chaque parcelle élémentaire, les densités varient continûment selon un dispositif inventé par Nelder : les lignes s'écartent de plus en plus en allant d'un côté à l'autre de la parcelle et sur chaque ligne, les arbres s'écartent de plus en plus en parcourant la ligne. Seuls 5 à 10 palmiers ont été mesurés par parcelle élémentaire.

Ici, on est en présence d'un essai Alpha-Lattice. Il y a des blocs incomplets qui sont regroupés en répétitions complètes. Normalement, il faudrait prendre les blocs en effet aléatoire. Cela permet d'introduire une corrélation entre les observations faites dans un même bloc et donc d'introduire une sctrucuration spatiale "grossière" dans le modèle. Ici, l'effet aléatoire bloc capte une grande partie de la corrélation spatiale intra-block. Ainsi, dans un but pédagogique de prise en compte de la corrélation spatiale entre les observations, nous ne considérerons pas cet effet aléatoire, même si en théorie, il faudrait.

Le modèle statistique est alors le suivant :

$$Y_{ijk} = \mu + G_i + X_{ijk}\beta_0 + X_{ijk}\beta_i + u_j + \varepsilon_{ijk}$$

Où

- i est l'indice des croisements
- j est l'indice des parcelles
- k est l'indice des palmiers dans la parcelle
- Y_{ijk} est l'observation du diamètre de l'individu ijk
- μ est l'intercept
- G_i est l'effet fixe du croisement i
- X_{ijk} est la densité de plantation de l'individu ijk
- β_0 est l'effet de la densité de plantation sur le diamètre
- β_i est l'effet d'interaction de la densité de plantation sur le diamètre pour le $i^{\text{ème}}$ croissement
- u_j est l'effet aléatoire de la $j^{\text{ème}}$ parcelle, tel que le vecteur de l'ensemble des effets des parcelle $u \sim N(0, \sigma_u^2\mathbf{I})$
- ε_{ijk} le résidu associé à l'observation ijk, tel que le vecteur de l'ensemble des résidus $\varepsilon \sim N(0, R)$, avec R une matrice de covariance permettant de prendre en compte la dépendance spatiale entre les individus.

```r
# library(sf)
library(spdep)
library(sp)
library(gstat)
```

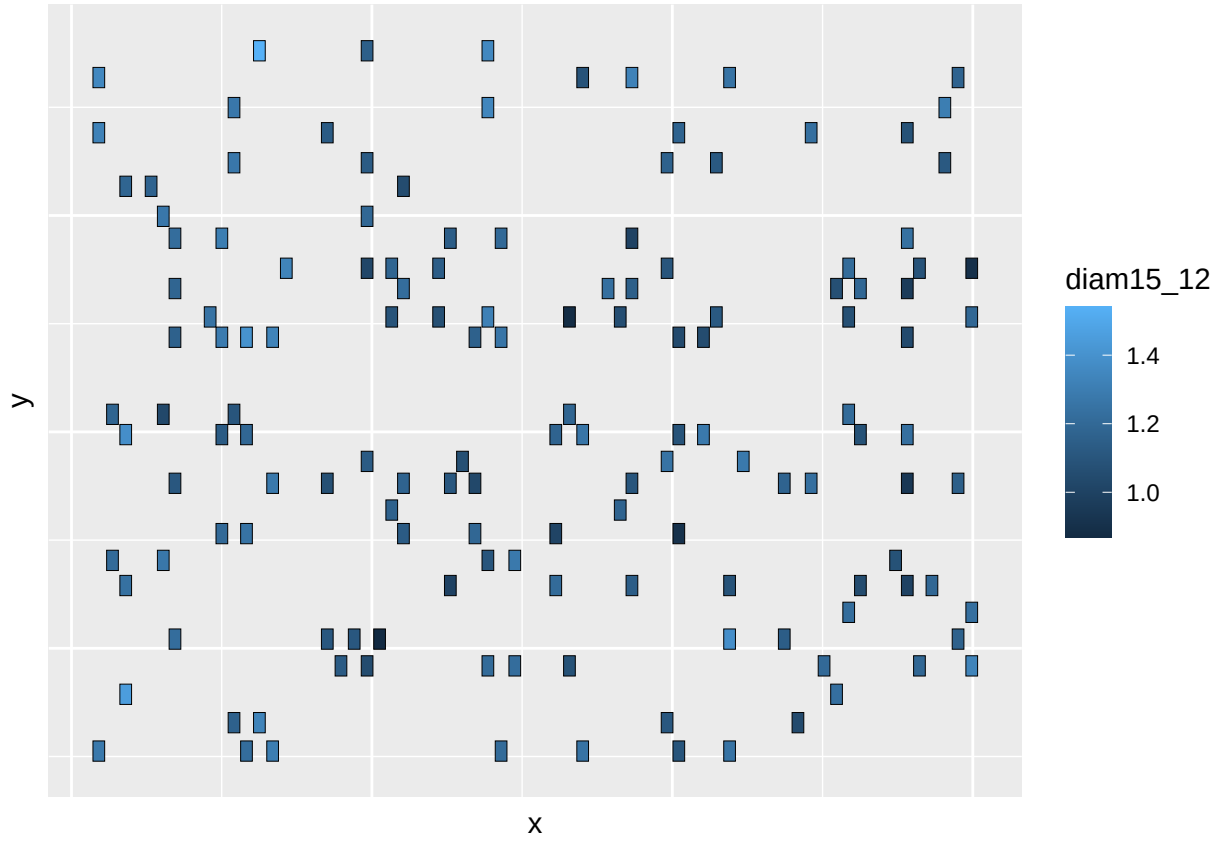
Importation des données

```r
toutveg <- read.csv("Data/Données_spatiales_palmier/donnees_essai_densite_bresil.csv")
donobs <- subset(toutveg, nature=="U" & sigpart=="" & densite>=103 & !is.na(diam15_12))
dim(donobs)
```

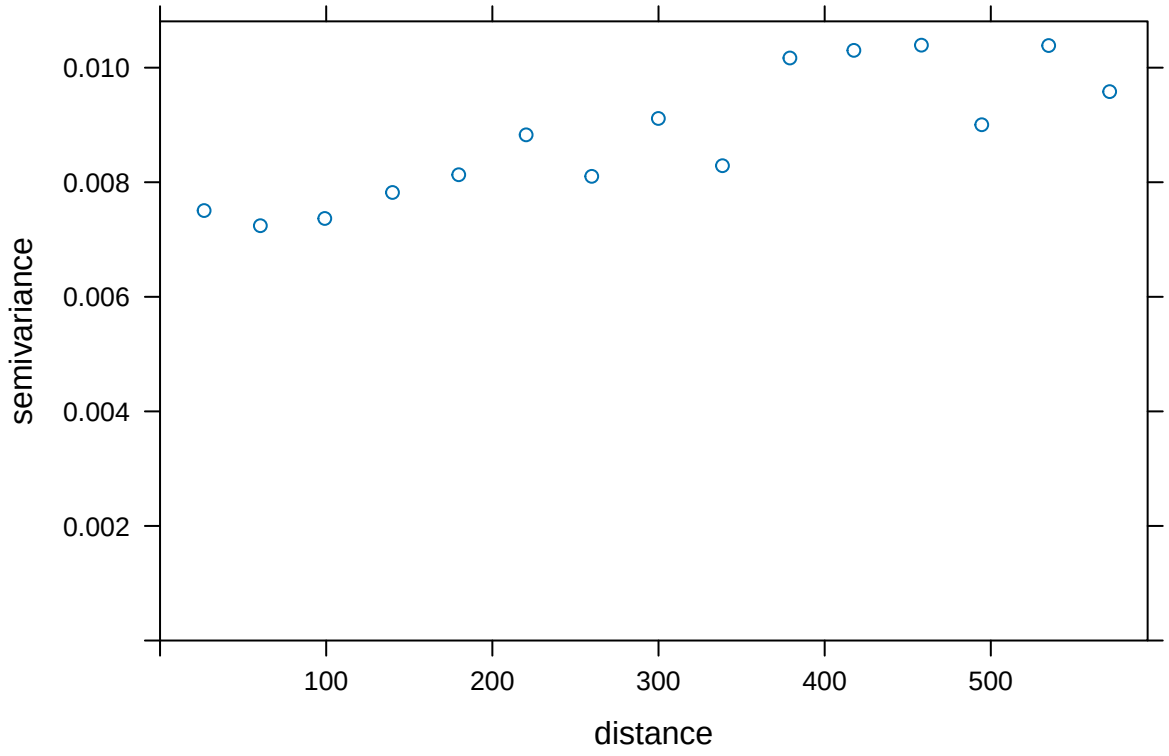
```
## [1] 145  51
```

```r
# Declaration des facteurs
donobs$numord_f <- as.factor(donobs$numord)
```

```r
library(tidyverse)
ggplot(donobs, aes(x=x, y=y)) +
  # facet_grid(block~., labeller = label_both) +
  geom_tile(aes(fill=diam15_12), color = "black") +
  # scale_fill_brewer(palette="Pastel1") +
  # geom_tileborder(aes(group=1, grp=trt), lwd=1.5) +
  # geom_text(aes(label = trt), color="black", size = 4) +
  # scale_color_brewer(palette="RdGy")+
  # ggtitle("BIB design") +
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank()
  )
```




```
# Ajustements avec gstat
# Declaration des coordonnees
donobs2 <- donobs
sp::coordinates(donobs2) = ~x+y

# Creation du variogramme empirique avec les effets fixes, la distance max est d'environ 600m
vemp0 <- gstat::variogram(diam15_12~numord_f+densite+densite %in% numord_f, data=donobs2, cutoff=600)
# trace du variocgramme empirique
plot(vemp0)
```



4.1 nlme

Dans ce modèle, nous venons réaliser un modèle mixte avec gls afin de ne pas inclure d'effets aléatoire mais inclure une corrélation spatiale.

```
mod_nlme_spher <- gls(
  diam15_12 ~ numord_f + densite + densite:numord_f,
            correlation = corSpatial(form = ~ x + y, type = "spherical"),
            data = donobs)

mod_nlme_exp <- gls(
  diam15_12 ~ numord_f + densite + densite:numord_f,
            correlation = corExp(form = ~ x + y, value =20, fixed = TRUE),
            data = donobs)

AIC(mod_nlme_spher, mod_nlme_exp)
```

```
##                df       AIC
## mod_nlme_spher 18 -117.1891
## mod_nlme_exp   17 -108.8685
```

```
# summary(mod_nlme_spher)

residuals_mod_nlme <- residuals(mod_nlme_spher)
```

```r
coords <- donobs[, c("x", "y")]
# dist_matrix <- as.matrix(dist(coords))
nb <- dnearneigh(coords, 0, 100)
listw <- nb2listw(nb)
moran.test(residuals(mod_nlme_spher), listw)
```

```
##
##  Moran I test under randomisation
##
## data:  residuals(mod_nlme_spher)
## weights: listw
##
## Moran I statistic standard deviate = 5.528, p-value = 1.619e-08
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      0.0877722116     -0.0069444444      0.0002935719
```

```r
predicted_values <- predict(mod_nlme_spher)
observed_values <- donobs$diam15_12

SST <- sum((observed_values - mean(observed_values))^2)
SSE <- sum((observed_values - predicted_values)^2)
R2_pred <- 1 - (SSE / SST)
print(R2_pred)
```

```
## [1] 0.3285931
```

4.2 sommer

Avec le package **Sommer**, on vient intégrer la corrélation spatiale dans les effets aléatoires par l'argument `spl2Da` en précisant les coordonnées pour x et y. La prise en compte de la corrélation spatiale se fait à l'aide de splines à 2 dimensions

```r
donobs$Bloc <- as.factor(donobs$Bloc)

mod_sommer <- mmer(
  fixed = diam15_12 ~ numord_f + densite + densite:numord_f,
  random = ~ spl2Da(x.coord = x, y.coord= y),
  data = donobs, verbose = FALSE, dateWarning = FALSE, nIters = 3
)

# summary(mod_sommer)

residuals_mod_sommer <- mod_sommer$residuals
coords <- donobs[, c("x", "y")]
dist_matrix <- as.matrix(dist(coords))
nb <- dnearneigh(coords, 0, 100)
listw <- nb2listw(nb)
moran.test(residuals_mod_sommer, listw)
```

```
##
##  Moran I test under randomisation
##
## data:  residuals_mod_sommer
## weights: listw
##
## Moran I statistic standard deviate = 5.8992, p-value = 1.826e-09
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      0.0941663452     -0.0069444444      0.0002937682
```

```
predicted_values <- fitted(mod_sommer)$dataWithFitted$diam15_12.fitted
```

```
## Version out of date. Please update sommer to the newest version using:
## install.packages('sommer') in a new session
##  Use the 'dateWarning' argument to disable the warning message.Version out of date. Please update so
## install.packages('sommer') in a new session
##  Use the 'dateWarning' argument to disable the warning message.
```

```
observed_values <- donobs$diam15_12

SST <- sum((observed_values - mean(observed_values))^2)
SSE <- sum((observed_values - predicted_values)^2)
R2_pred <- 1 - (SSE / SST)

print(R2_pred)
```

```
## [1] 0.5188692
```

4.3 BRMS

Avec le package **BRMS**, il est possible d'intégrer une corrélation spatiale.

On vient créer un objet spatial en utilisant les coordonnées 'x' et 'y' du jeu de données. Ensuite, on vient créer une liste de voisins en utilisant une distance minimale de 0 et une distance maximale de 50 unités. Chaque point est lié à ses voisins dans cette plage de distance. On finit par la convertir en une liste de poids spatiaux.

```
donobs_sf <- st_as_sf(donobs, coords = c("x", "y"), crs = 4326)
coords <- st_coordinates(donobs_sf)
neigh <- dnearneigh(coords, 0, 50)
M <- nb2listw(neigh)

fit_brms <- brms::brm(
  bf(diam15_12 ~  1 + numord_f + densite + densite:numord_f + sar(M, type = "error")),
  family = gaussian(),
  data = donobs,
  data2 = list(M=M),
  # prior = priors,
  iter=5000, thin = 5, cores=4, chains=4)
```

```
## Compiling Stan program...
```

```
## Start sampling
```

```
# summary(fit_brms)

residuals_fit_brms <- residuals(fit_brms)[,1]
moran.test(residuals_fit_brms, M)
```

```
##
##  Moran I test under randomisation
##
## data:  residuals_fit_brms
## weights: M
##
## Moran I statistic standard deviate = 4.6613, p-value = 1.571e-06
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.153464274      -0.006944444       0.001184230
```

```
predicted_values <- predict(fit_brms)[,1]
observed_values <- donobs$diam15_12

SST <- sum((observed_values - mean(observed_values))^2)
SSE <- sum((observed_values - predicted_values)^2)
R2_pred <- 1 - (SSE / SST)
print(R2_pred)
```

```
## [1] 0.3064366
```

4.4 BGLR

Avec le package **BGLR**, il est possible d'intégrer une corrélation spatiale en créant une matrice de distance qui sera intégrée dans la partie aléatoire du modèle.

```
# Distance Matrix
# # noyaux gaussien
# dist_mat<-as.matrix(dist(donobs[, c("x", "y")], method="euclidean"))^2
# a<-mean(dist_mat)

# noyaux exponentien = spatial AR
dist_mat<-as.matrix(dist(donobs[, c("x", "y")], method="euclidean"))
a = 20
rho = exp(-1/a); print(rho)
```

```
## [1] 0.9512294
```

```r
# KERNEL
K_sp<-exp(-dist_mat/a)


donobs$numord_f = factor(donobs$numord_f)
ETA = list(
  fixed=list(~ 1+numord_f + densite + numord_f:densite, data=donobs, model='FIXED'),
  # block = list(K=tcrossprod(design.matrix(donobs$Bloc)), model = "RKHS", data = donobs),
  sp = list(K = K_sp, model = "RKHS") # cor spatiale
)

dir.create("fit_BGLR")
```

```
## Warning in dir.create("fit_BGLR"): 'fit_BGLR' existe déjà
```

```r
fit_BGLR = BGLR(y = donobs$diam15_12, ETA = ETA,
                nIter =5000, burnIn =2000, saveAt ="fit_BGLR/", verbose =FALSE)

residuals_fit_BGLR <- residuals(fit_BGLR)
coords <- donobs[, c("x", "y")]
# dist_matrix <- as.matrix(dist(coords))
nb <- dnearneigh(coords, 0, 100)
listw <- nb2listw(nb)
moran.test(residuals_fit_BGLR, listw)
```

```
##
##  Moran I test under randomisation
##
## data:  residuals_fit_BGLR
## weights: listw
##
## Moran I statistic standard deviate = -0.032056, p-value = 0.5128
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##     -0.0074940570     -0.0069444444      0.0002939638
```

```r
predicted_values <- predict(fit_BGLR)
observed_values <- donobs$diam15_12

SST <- sum((observed_values - mean(observed_values))^2)
SSE <- sum((observed_values - predicted_values)^2)
R2_pred <- 1 - (SSE / SST)

print(R2_pred)
```

```
## [1] 0.6798809
```

Pour récupérer les différentes variances, il faut exécuter les lignes de code ci-dessous :

```
# variance résiduelle
fit_BGLR$varE
```

```
## [1] 0.006504185
```

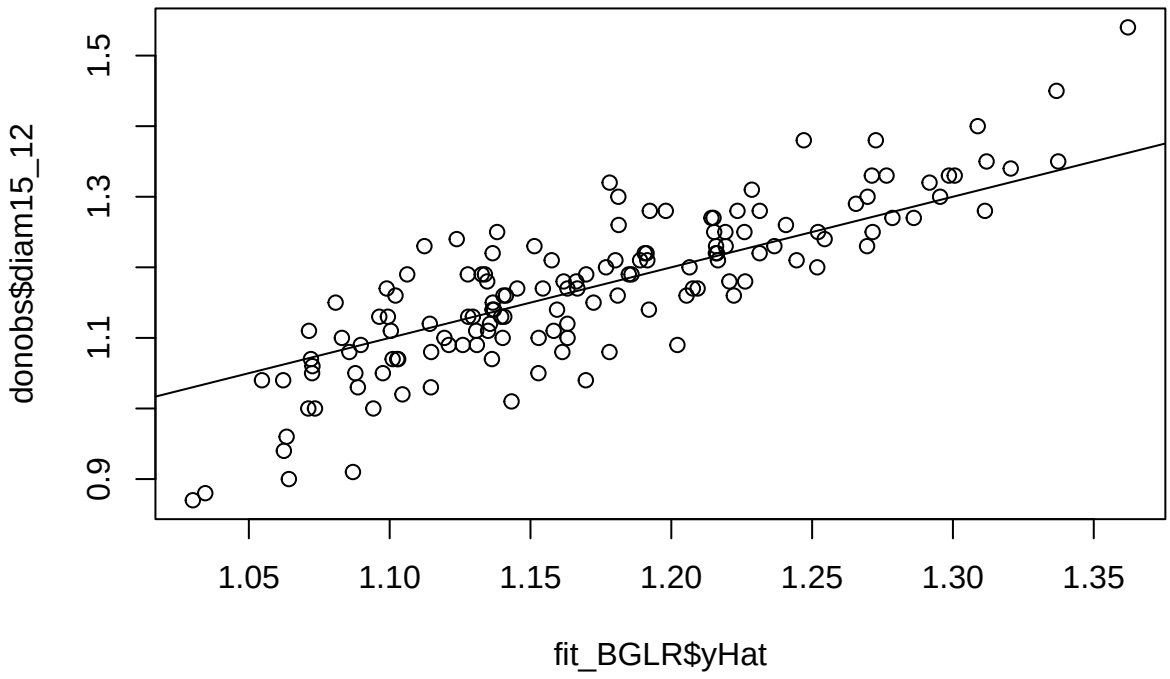
```
# Effet fixe
fit_BGLR$ETA$fixed$b
```

```
##          numord_f5         numord_f11         numord_f14         numord_f15
##       0.0968401867       0.2959171251       0.2426691274      -0.1534117114
##         numord_f18         numord_f19         numord_f20            densite
##       0.5170330392      -0.2520977321       0.2441572875      -0.0033656758
##  numord_f5:densite numord_f11:densite numord_f14:densite numord_f15:densite
##      -0.0001261752      -0.0014836643      -0.0016937698       0.0023947528
## numord_f18:densite numord_f19:densite numord_f20:densite
##      -0.0031501063       0.0021328783      -0.0011480582
```

```
# Effets aléatoires
# fit_BGLR$ETA$block$varU
fit_BGLR$ETA$sp$varU
```

```
## [1] 0.003888744
```

```
plot(fit_BGLR$yHat, donobs$diam15_12); abline(0, 1)
```



```r
cor(fit_BGLR$yHat, donobs$diam15_12)^2
```

```
## [1] 0.7211463
```

---

5 Simulation : cas du modèle animal

Le but de ces simulations est de pouvoir comparer les performances des différentes librairies.

Les simulations suivantes sont basées sur une matrice d'apparentement (A) issue de données réelles sur le palmier à huile (voir exemple 1). Ce jeu de données comprend 1284 individus dont 1227 sont des descendants et 57 sont des géniteurs. Les descendants vont servir de jeu de données d'entraînement tandis que les géniteurs vont servir de jeu de données de validation.

```r
###### Donnees de production moyenne 3-5 ans
df <- read.csv2("Data/Données_palmier_multi_essais/prod_3_5.csv")
###### Donnees des pédigrés
ped <- read.csv2("Data/Données_palmier_multi_essais/ped_3_5.csv")
ped <- nadiv::prepPed(ped)
```

```r
df_1 = df %>% filter(trial == "ALGP01")

df_1$treef_id = as.factor(as.character(df_1$treef_id))

## Préparation matrice d'apparentement
descendants_names = as.character(df_1$treef_id)

ped1 <- prunePed(ped, phenotyped = descendants_names)

genitor_names = unique(as.character(ped1$treef_id[!ped1$treef_id %in% descendants_names]))

## Calcul de la matrice de Relationship (Wright, 1922) qui est le double de la
## matrice de coefficients de parenté
A <- 2*kinship2::kinship(id = ped1$treef_id, momid = ped1$mother_id, dadid = ped1$father_id)

A = A[c(descendants_names, genitor_names), c(descendants_names, genitor_names)]


A11 = A[descendants_names, descendants_names]
A21=A[genitor_names, descendants_names]
```

5.1 Simulation de données

Nous simulons des données à l'aide d'un modèle animal avec une variance génétique de 3 et une variance résiduelle de 1. L'intercept est fixé à 10 :

$$Y = 10 + u + \varepsilon, u \sim N(0, 3A), \varepsilon \sim N(0, 1I)$$

```r
n = nrow(A)
su2 = 3
set.seed(1234)
u = c(mvnfast::rmvn(1, rep(0, n), su2 * A))

se2 = 1
set.seed(34577)
e = c(mvnfast::rmvn(1, rep(0, n), su2 * diag(n)))
mu=10


df_sim = data.frame(Y = mu + u + e,
                    id = rownames(A),
                    u = u
)

rownames(df_sim) = df_sim$id
df_sim$Y_na_gen = df_sim$Y
df_sim[genitor_names, "Y_na_gen"] = NA


df_sim_desc = df_sim %>% filter(id %in% descendants_names)
df_sim_gen = df_sim %>% filter(id %in% genitor_names)
```

5.2 Analyse

Certaines librairies permettent l'imputation des données manquantes dans la variable réponse. Cela peut-être utilisé pour faire la prédiction des géniteurs en travaillant sur l'ensemble des données et en mettant des NA pour la variable réponse des géniteurs. Pour les librairies qui le permettent, nous comparerons cette approche avec les BLUPs.

5.2.1 lme4GS

Avec lme4GS il n'est pas possible de faire la prédiction des géniteurs à partir de l'imputation des NA

Prédiction des BLUPs des géniteurs à partir des équations d'Anderson

```r
library(lme4GS)

fit_lme4gs = lmerUvcov(Y ~ 1  + (1|id),
                       data = df_sim_desc,
                       Uvcov=list(id=list(K=A11)),
                       verbose = FALSE)

sum_lme4GS = summary(fit_lme4gs)
sum_lme4GS
```

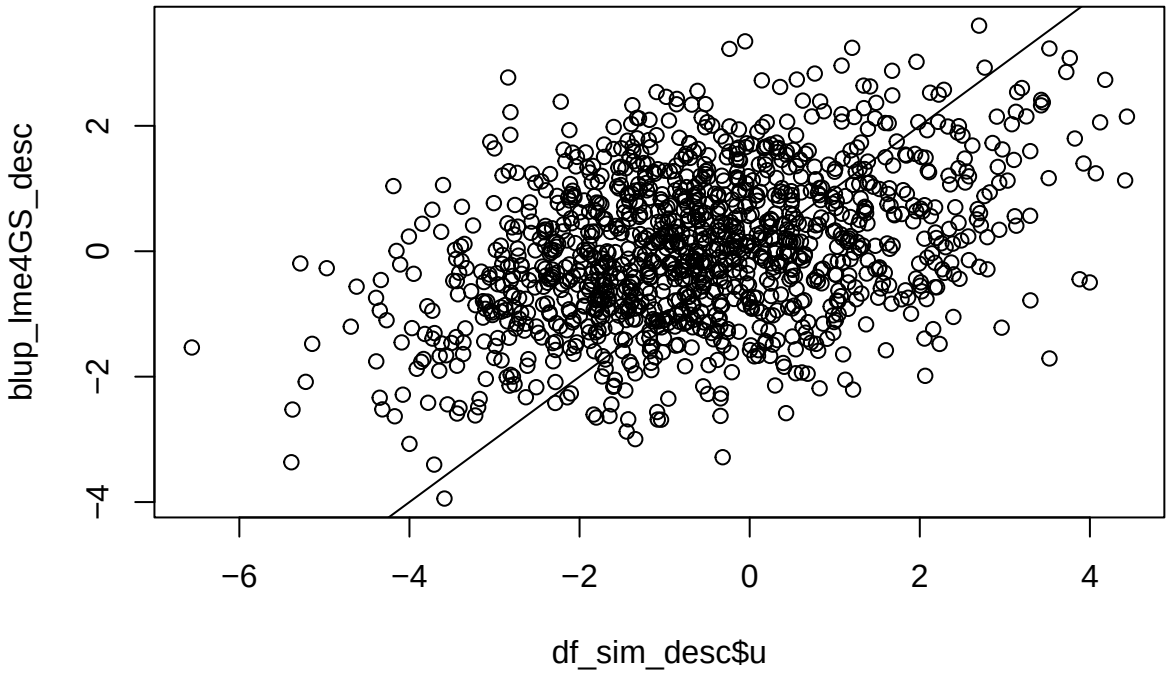
```
## Linear mixed model fit by REML ['lmerUvcov']
## Formula: Y ~ 1 + (1 | id)
##    Data: df_sim_desc
##
## REML criterion at convergence: 5207
```

```
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.7253 -0.5558 -0.0274  0.5845  2.4763
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  id       (Intercept) 4.081    2.020
##  Residual             2.542    1.594
## Number of obs: 1227, groups:  id, 1227
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   9.2919     0.8562   10.85
```

```
## R2 sur descendants
cor(predict(fit_lme4gs), df_sim_desc$Y)^2
```

```
## [1] 0.7757884
```

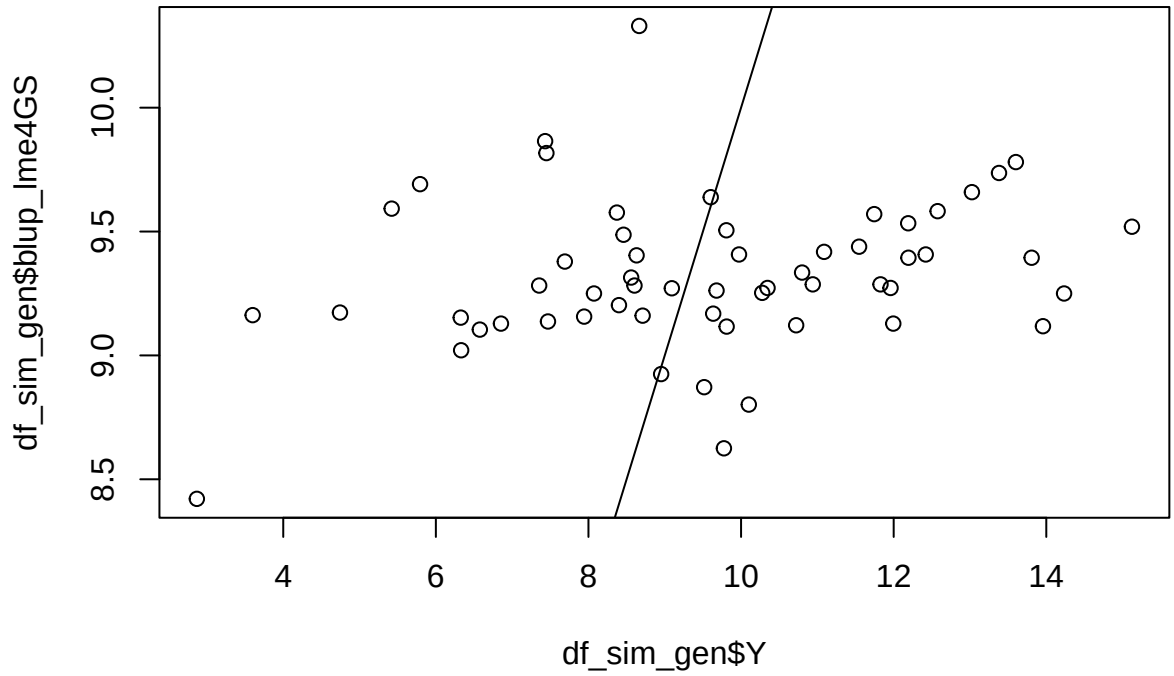
```
## R2 sur géniteurs
blup_lme4GS_desc = ranef(fit_lme4gs)[[1]][, 1]
plot(df_sim_desc$u, blup_lme4GS_desc); abline(0, 1)
```



```r
cor(df_sim_desc$u, blup_lme4GS_desc)
```

```
## [1] 0.3772676
```

```r
mu_est = sum_lme4GS$coefficients[1, 1]
df_sim_gen$blup_lme4GS = mu_est + as.vector(A21 %*% solve(A11) %*% blup_lme4GS_desc)

plot(df_sim_gen$Y, df_sim_gen$blup_lme4GS); abline(0, 1)
```




```r
R2_G_lme4GS = cor(df_sim_gen$Y, df_sim_gen$blup_lme4GS)^2
R2_G_lme4GS
```

```
## [1] 0.0630943
```

5.2.2 sommer

Avec sommer, il n'est pas possible de faire la prédiction des géniteurs à partir de l'imputation des NA

Prédiction des BLUPs des géniteurs à partir des équations d'Anderson

```r
library(sommer)
```

```r
fit_som <- mmer(Y ~ 1,
                random =  ~ vsr(id, Gu=A11),
                data=df_sim_desc, verbose = FALSE, dateWarning = FALSE, nIters = 10)

summary(fit_som)
```

```
## ============================================================
##          Multivariate Linear Mixed Model fit by REML
## **********************  sommer 4.3  **********************
## ============================================================
##         logLik      AIC      BIC Method Converge
## Value -391.058 784.1159 789.2283     NR     TRUE
## ============================================================
## Variance-Covariance components:
##           VarComp VarCompSE Zratio Constraint
## u:id.Y-Y    4.081    1.1498  3.549   Positive
## units.Y-Y   2.542    0.3999  6.356   Positive
## ============================================================
## Fixed effects:
##   Trait      Effect Estimate Std.Error t.value
## 1     Y (Intercept)    9.292    0.8562   10.85
## ============================================================
## Groups and observations:
##         Y
## u:id 1227
## ============================================================
## Use the '$' sign to access results and parameters
```

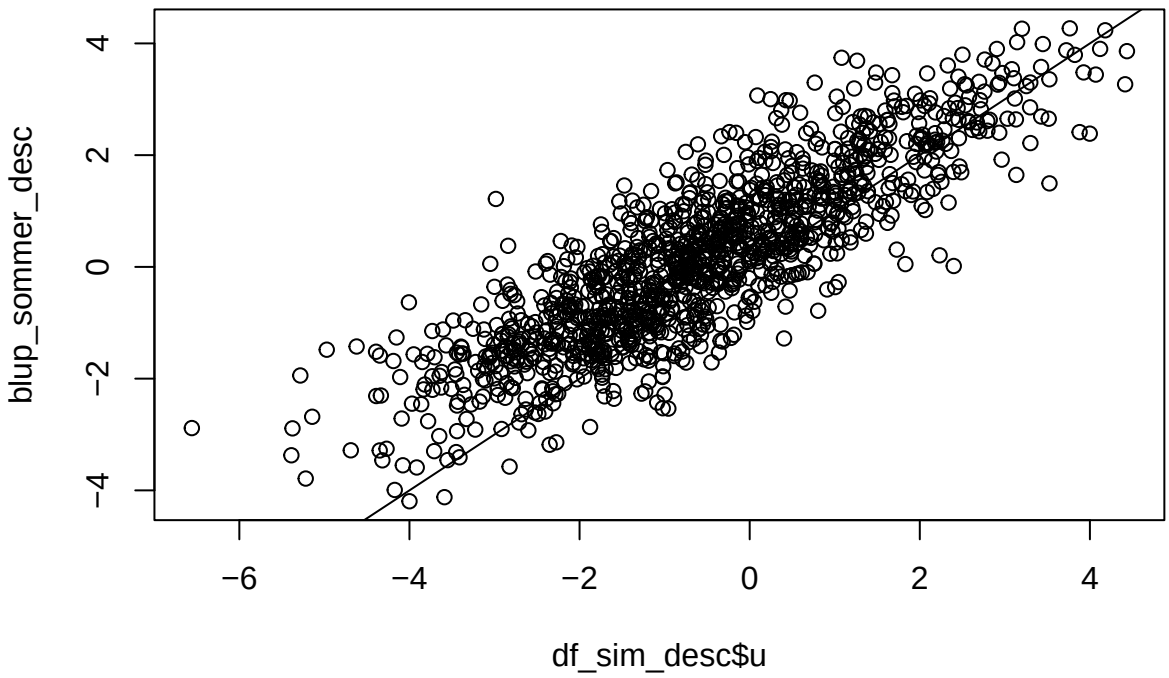
```r
## R2 sur descendants
output_fitted_sommur = fitted(fit_som)
```

```
## Version out of date. Please update sommer to the newest version using:
## install.packages('sommer') in a new session
##  Use the 'dateWarning' argument to disable the warning message.Version out of date. Please update so
## install.packages('sommer') in a new session
##  Use the 'dateWarning' argument to disable the warning message.
```

```r
cor(output_fitted_sommur$dataWithFitted$Y.fitted, df_sim_desc$Y)^2
```

```
## [1] 0.7757796
```

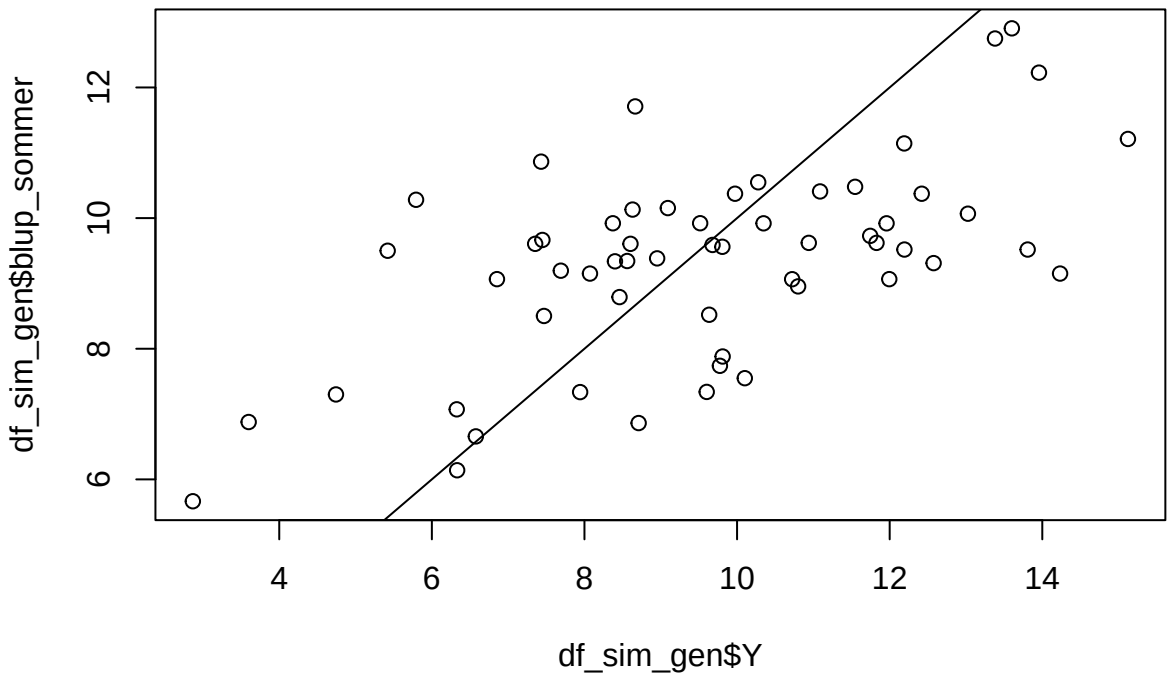
```r
## R2 sur géniteurs
blup_sommer_desc = randef(fit_som)$`u:id`$Y
plot(df_sim_desc$u, blup_sommer_desc); abline(0, 1)
```




```
cor(df_sim_desc$u, blup_sommer_desc)^2
```

```
## [1] 0.7308071
```

```
mu_est = coef(fit_som)$Estimate
df_sim_gen$blup_sommer = mu_est + as.vector(A21 %*% solve(A11) %*% blup_sommer_desc)

plot(df_sim_gen$Y, df_sim_gen$blup_sommer); abline(0, 1)
```




```
corsom = cor(df_sim_gen$Y, df_sim_gen$blup_sommer)^2
corsom
```

```
## [1] 0.3618946
```

```
## rank(df_sim_gen$Y)
## rank(df_sim_gen$blup_sommer)
```

5.2.3 BGLR

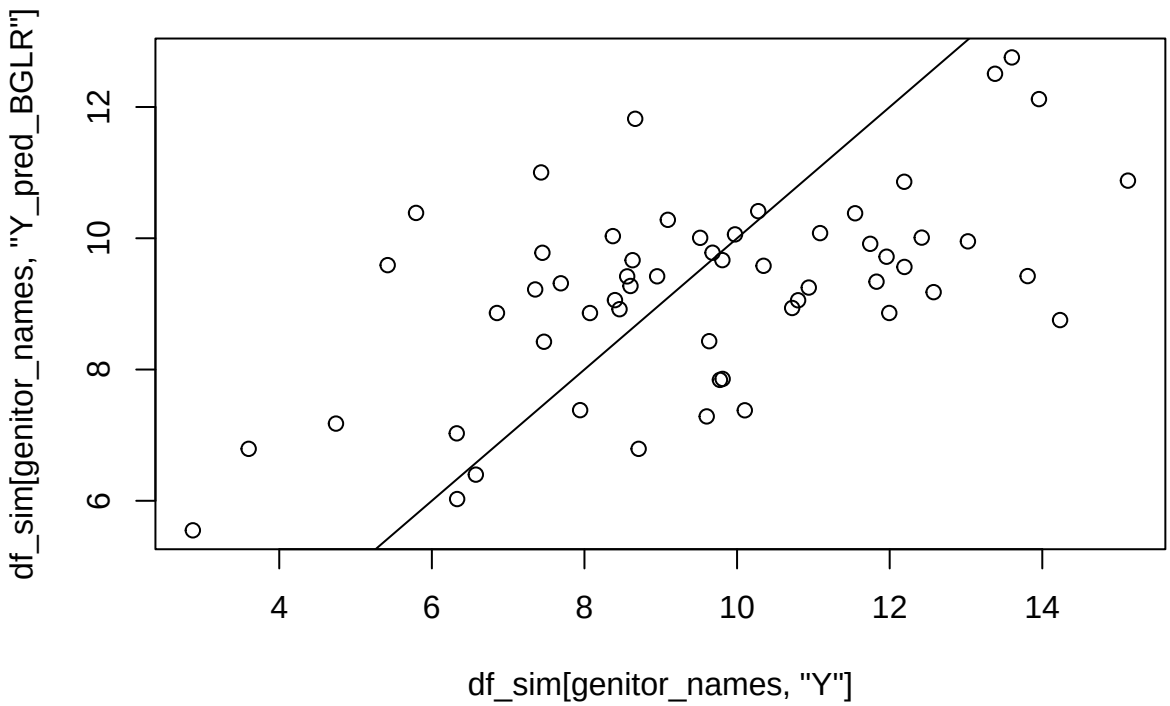
Prédiction des géniteurs par l'imputation des NA

```
library(BGLR)

dir.create("fit_BGLR")

## prediction des BLUPs des géniteurs à partir des du calcul manuel

ETA = list(
  u = list(K = A, model = "RKHS")
)

fit_BGLR = BGLR(y = df_sim$Y_na_gen, ETA = ETA,
                nIter =5000, burnIn =2000, saveAt ="fit_BGLR/")
```

```r
summary(fit_BGLR)
```

```
## --------------------> Summary of data & model <--------------------
##
##  Number of phenotypes= 1227
##  Min (TRN)=  2.432064
##  Max (TRN)=  16.34819
##  Variance of phenotypes (TRN)= 5.8449
##  Residual variance= 2.5496
##  N-TRN= 1227  N-TST= 57
##  Correlation TRN= 0.8801
##
##  -- Linear Predictor --
##
##  Intercept included by default
##  Coefficientes in ETA[ 1 ] ( u ) were assumed to be normally distributed with zero mean and
##  covariance (or its eigendecoposition) provided by user
##
## ----------------------------------------------------------------
```

```r
df_sim$Y_pred_BGLR = predict(fit_BGLR)

plot(df_sim[genitor_names, "Y"], df_sim[genitor_names, "Y_pred_BGLR"]); abline(0, 1)
```



```r
cor(df_sim[genitor_names, "Y"], df_sim[genitor_names, "Y_pred_BGLR"])^2
```

```
## [1] 0.3283239
```

Prédiction des BLUPs des géniteurs à partir des équations d'Anderson

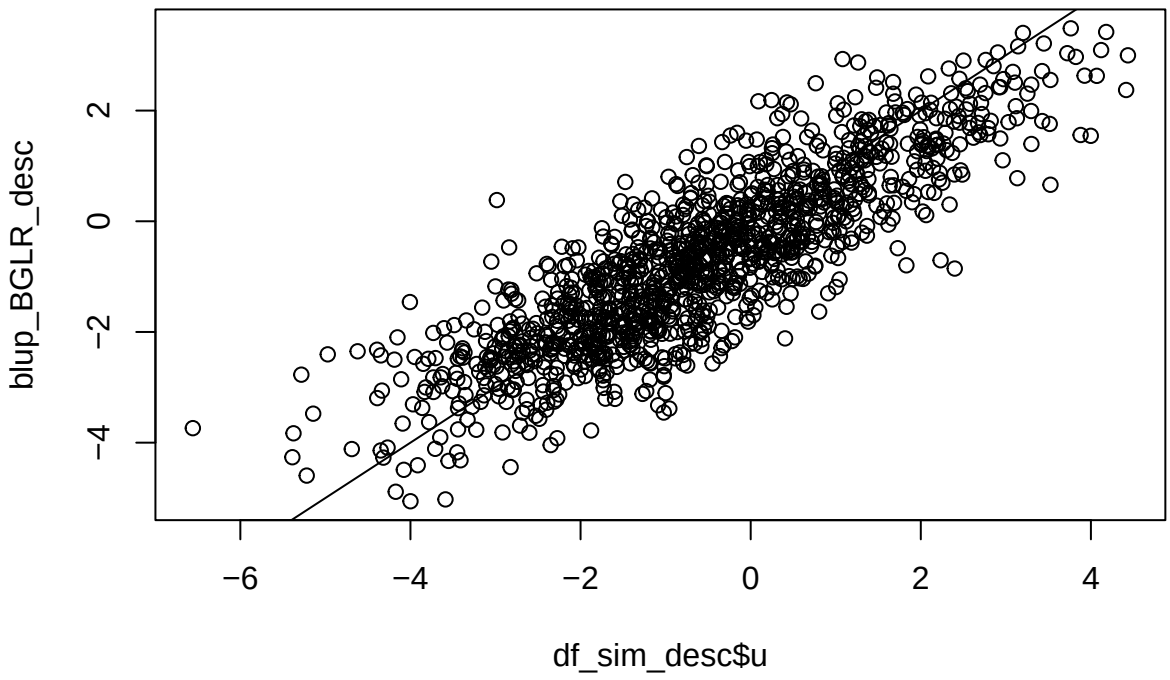
```r
library(BGLR)

dir.create("fit_BGLR")

ETA = list(
  u = list(K = A11, model = "RKHS")
)

fit_BGLR = BGLR(y = df_sim_desc$Y, ETA = ETA,
                nIter =5000, burnIn =2000, saveAt ="fit_BGLR/")

summary(fit_BGLR)
```

```
## --------------------> Summary of data & model <--------------------
##
##  Number of phenotypes= 1227
##  Min (TRN)=  2.432064
##  Max (TRN)=  16.34819
##  Variance of phenotypes (TRN)= 5.8449
##  Residual variance= 2.5418
##  N-TRN= 1227   N-TST=0
##
##
##  -- Linear Predictor --
##
##  Intercept included by default
##  Coefficientes in ETA[ 1 ] ( u ) were assumed to be normally distributed with zero mean and
##  covariance (or its eigendecoposition) provided by user
##
## ------------------------------------------------------------------
```

```r
varB <- fit_BGLR$ETA$u$varU
varB_sd <- fit_BGLR$varE

## R2 descendants :
cor(predict(fit_BGLR), df_sim_desc$Y)^2
```

```
## [1] 0.7808054
```

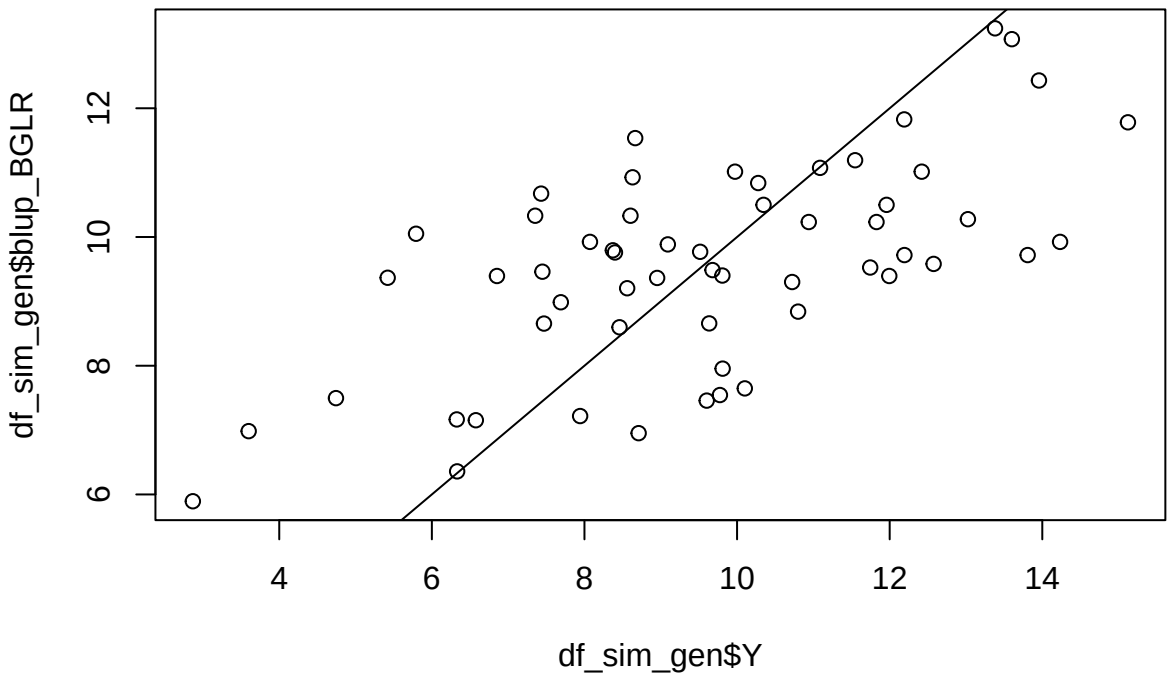
```r
## BLUP descendants
blup_BGLR_desc = fit_BGLR$ETA$u$u
plot(df_sim_desc$u, blup_BGLR_desc); abline(0, 1)
```




```
cor(df_sim_desc$u, blup_BGLR_desc)
```

```
## [1] 0.8539859
```

```
## R2 géniteur
mu_est = fit_BGLR$mu
df_sim_gen$blup_BGLR = mu_est + as.vector(A21 %*% solve(A11) %*% blup_BGLR_desc)

plot(df_sim_gen$Y, df_sim_gen$blup_BGLR); abline(0, 1)
```




```
corbglr = cor(df_sim_gen$Y, df_sim_gen$blup_BGLR)^2
corbglr
```

```
## [1] 0.4034351
```

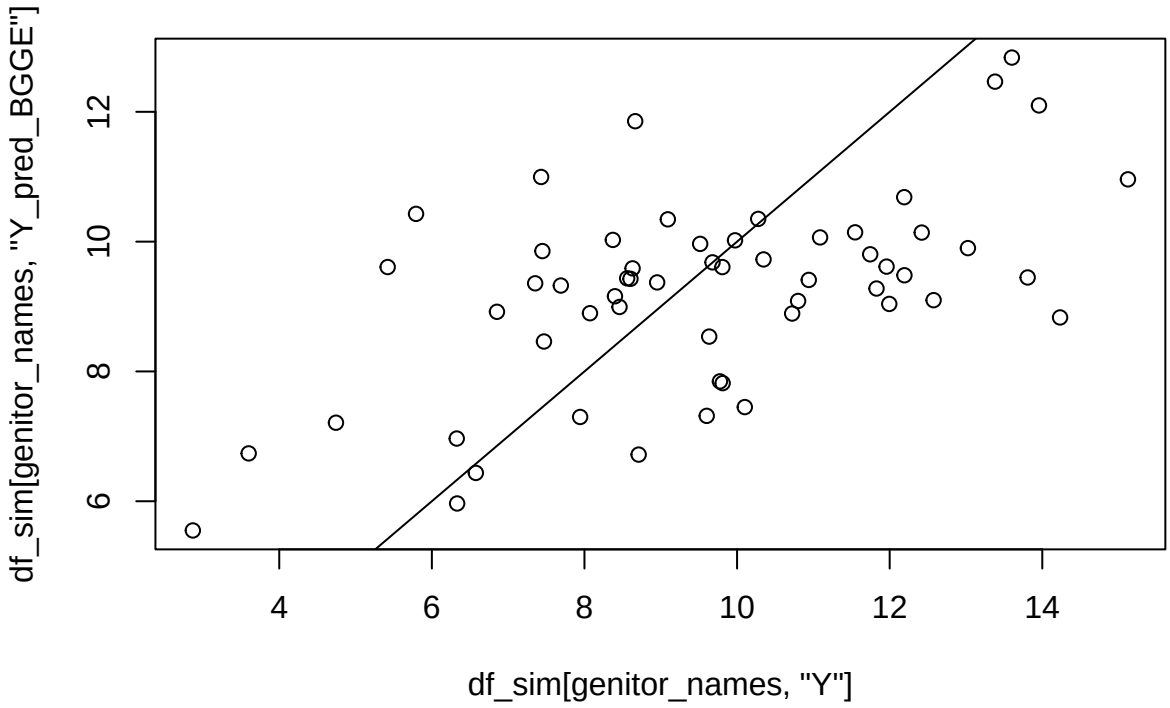
```
## rank(df_sim_gen$Y)
## rank(df_sim_gen$blup_sommer)
```

5.2.4 BGGE

Prédiction des géniteurs par l'imputation des NA

```
library(BGGE)

K = list(
  u = list(Kernel = A, Type = "D")
)

fit_BGGE = BGGE(y = df_sim$Y_na_gen,
                K = K,
                XF = NULL,
                ne=as.vector(c(nrow(df_sim_desc))),
                ite = 5000, burn = 2000, thin = 3)
```

```r
df_sim$Y_pred_BGGE = fit_BGGE$yHat

plot(df_sim[genitor_names, "Y"], df_sim[genitor_names, "Y_pred_BGGE"]); abline(0, 1)
```




```r
cor(df_sim[genitor_names, "Y"], df_sim[genitor_names, "Y_pred_BGGE"])^2
```

```
## [1] 0.324317
```

Prédiction des BLUPs des géniteurs à partir des équations d'Anderson

```r
K = list(
  u = list(Kernel = A11, Type = "D")
)

fit_BGGE = BGGE(y = df_sim_desc$Y,
                K = K,
                XF = NULL,
                ne=as.vector(c(nrow(df_sim_desc))),
                ite = 5000, burn = 2000, thin = 3)

summary(fit_BGGE)
```

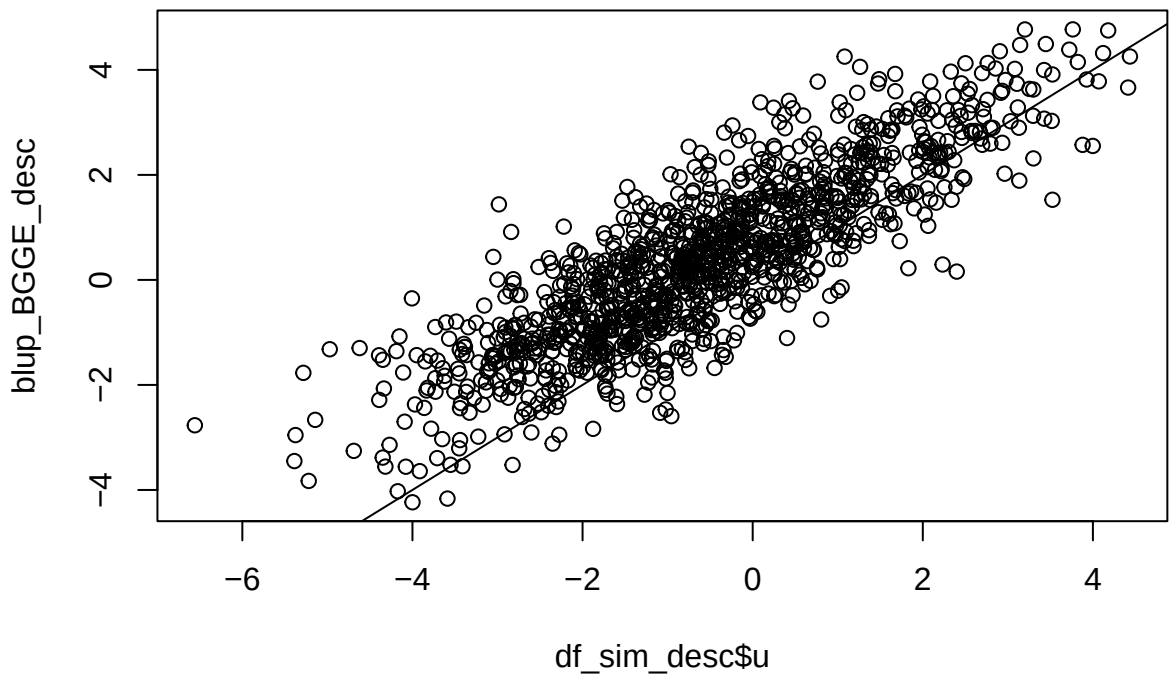
```
##         Length Class  Mode
## yHat    1227   -none- numeric
```

```
## varE       1   -none- numeric
## varE.sd    1   -none- numeric
## K          1   -none- list
## chain      3   -none- list
```

```
var_u <- fit_BGGE$K$u$varu
var_res <- fit_BGGE$varE

## R2 descendant
cor(fit_BGGE$yHat, df_sim_desc$Y)^2
```

```
## [1] 0.8287371
```

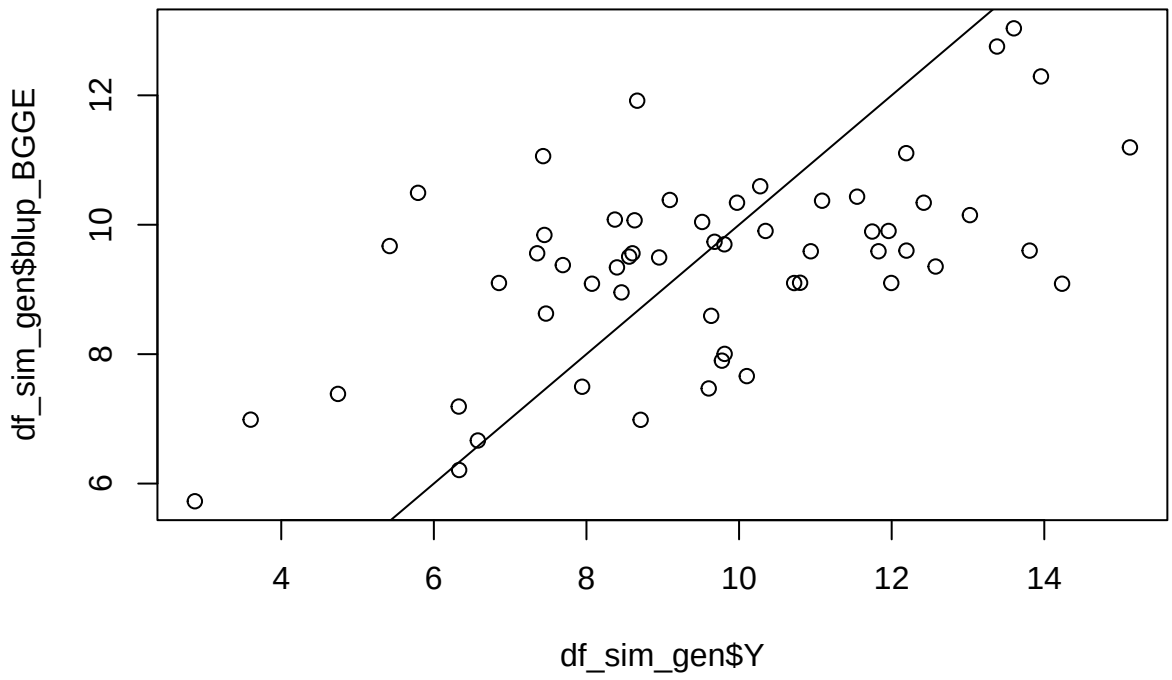
```
blup_BGGE_desc = fit_BGGE$K$u$u
plot(df_sim_desc$u, blup_BGGE_desc); abline(0, 1)
```




```
cor(df_sim_desc$u, blup_BGGE_desc)
```

```
## [1] 0.8463807
```

```
mu_est = mean(fit_BGGE$chain$mu)
df_sim_gen$blup_BGGE = mu_est + as.vector(A21 %*% solve(A11) %*% blup_BGGE_desc)

plot(df_sim_gen$Y, df_sim_gen$blup_BGGE); abline(0, 1)
```




```
corbgge = cor(df_sim_gen$Y, df_sim_gen$blup_BGGE)^2
corbgge
```

```
## [1] 0.3437796
```

5.2.5 brms

Prédiction des géniteurs par l'imputation des NA

```
library(brms)

fit_brms <- brms::brm(
  bf(Y_na_gen ~ 1 + (1|gr(id, cov=A))),
  family = gaussian(),
  data = df_sim,
  data2 = list(A=A),
  ## prior = priors,
  iter=5000, thin = 5, cores=4, chains=4)


df_sim$Y_pred_brms = predict(fit_brms)[, 1]

plot(df_sim[genitor_names, "Y"], df_sim[genitor_names, "Y_pred_brms"]); abline(0, 1)
cor(df_sim[genitor_names, "Y"], df_sim[genitor_names, "Y_pred_brms"])^2
```

Prédiction des BLUPs des géniteurs à partir des équations d'Anderson

```
library(brms)

fit_brms <- brms::brm(
  bf(Y ~ 1 + (1|gr(id, cov=A11))),
  family = gaussian(),
  data = df_sim_desc,
  data2 = list(A11=A11),
  ## prior = priors,
  iter=5000, thin = 5, cores=4, chains=4)

summary(fit_brms)
1.66^2
1.29^2

## R2 descendant
cor(predict(fit_brms)[,1], df_sim_desc$Y)^2


## BLUP descendant
blup_brms_desc = ranef(fit_brms)$id[, , 1][, 1]
plot(df_sim_desc$u, blup_brms_desc); abline(0, 1)
cor(df_sim_desc$u, blup_brms_desc)

tmp = summary(fit_brms)
mu_est = tmp$fixed$Estimate
df_sim_gen$blup_brms = mu_est + as.vector(A21 %*% solve(A11) %*% blup_brms_desc)

plot(df_sim_gen$Y, df_sim_gen$blup_brms); abline(0, 1)
corbrms = cor(df_sim_gen$Y, df_sim_gen$blup_brms)^2
corbrms
```

5.2.6 INLA

Calcul de l'inverse de la matrice d'apparentement au format "sparse". Utilisation du package "pedigreemm".

```
names(ped1) = c("Individual", "Parent1", "Parent2")

ped1$Individual = as.character(ped1$Individual)
ped1$Parent1 = as.character(ped1$Parent1)
ped1$Parent2 = as.character(ped1$Parent2)

library(pedigreemm)
ped1_inla = pedigreemm::pedigree(sire = ped1$Parent2, dam = ped1$Parent1, label = ped1$Individual)

T_Inv <- as(ped1_inla, "sparseMatrix")
## returns a sparse, unit lower-triangular matrix which is the inverse of the "L"
## part of the "LDL'" form of the Cholesky factorization of the relationship matrix.
## All non-zero elements below the diagonal are -0.5.
D_Inv <- Matrix::Diagonal(x=1/pedigreemm::Dmat(ped1_inla))
Ainv_pedigreemm <- t(T_Inv) %*% D_Inv %*% T_Inv
dimnames(Ainv_pedigreemm)[[1]] <- dimnames(Ainv_pedigreemm)[[2]] <- ped1_inla@label
```

```r
map = data.frame(Individual = ped1_inla@label, newId = 1:length(ped1_inla@label))
map$newId = as.numeric(map$newId)
```

Prédiction des géniteurs par l'imputation des NA

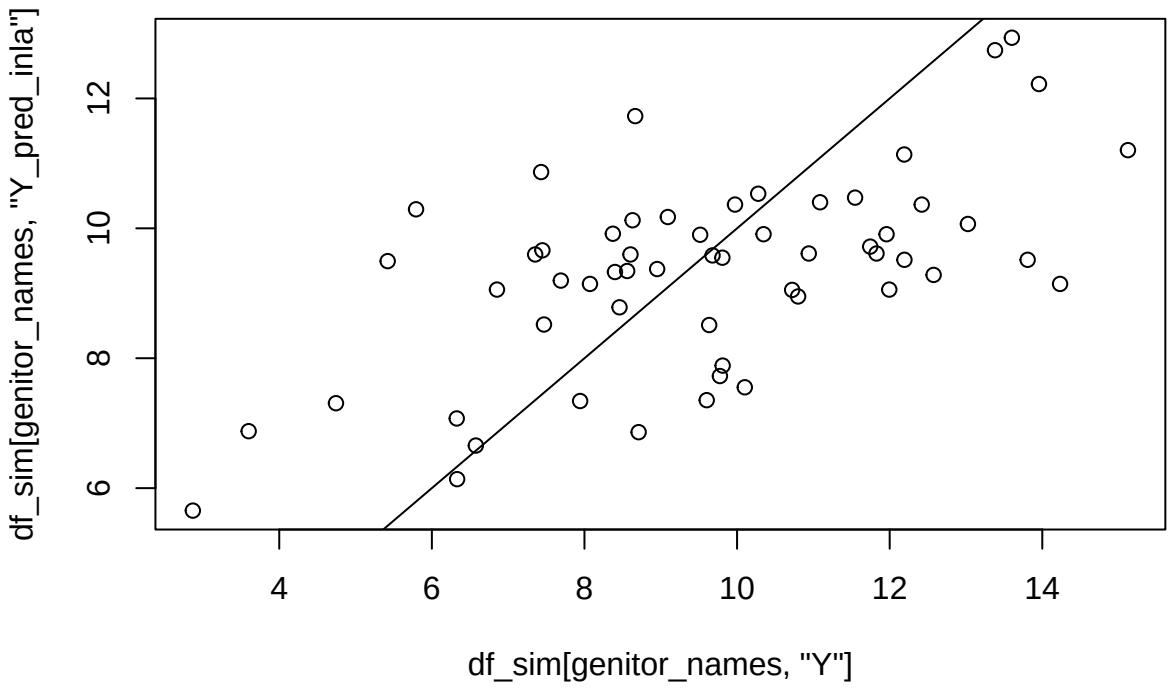
```r
df_sim$Individual = as.character(df_sim$id)
df_sim = left_join(df_sim, map)
rownames(df_sim) = df_sim$id
```

```r
library(INLA)

vtot <- var(df_sim_desc$Y, na.rm=TRUE)
s2max <- 10*vtot
prec.Random <- list(prior="pc.prec", param=c(s2max,0.01), fixed=FALSE)
prec.A <- list(prior="pc.prec", param=c(s2max, 0.01), fixed=FALSE)



fit_inla <- inla(Y_na_gen ~ 1 +
                   f(newId, model="generic0", Cmatrix=Ainv_pedigreemm, constr=FALSE, hyper=list(theta=prec.
                 data=df_sim,
                 family="gaussian",
                 control.family=list(hyper=list(theta=prec.Random)),
                 control.compute=list(dic=TRUE)
)

summary(fit_inla)
```

```
## Time used:
##     Pre = 2.85, Running = 3.07, Post = 0.365, Total = 6.29
## Fixed effects:
##              mean    sd 0.025quant 0.5quant 0.975quant  mode kld
## (Intercept) 9.286 0.913      7.481    9.288     11.083 9.288   0
##
## Random effects:
##   Name     Model
##     newId Generic0 model
##
## Model hyperparameters:
##                                          mean    sd 0.025quant 0.5quant
## Precision for the Gaussian observations 0.420 0.075      0.293    0.413
## Precision for newId                     0.234 0.067      0.129    0.226
##                                         0.975quant  mode
## Precision for the Gaussian observations      0.585 0.400
## Precision for newId                          0.392 0.209
##
## Deviance Information Criterion (DIC) ...............: 4975.89
## Deviance Information Criterion (DIC, saturated) ....: 1690.06
## Effective number of parameters .....................: 461.23
##
## Marginal log-Likelihood:  -3330.98
##  is computed
```

```
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```
df_sim$Y_pred_inla = fit_inla$summary.fitted.values$mean

plot(df_sim[genitor_names, "Y"], df_sim[genitor_names, "Y_pred_inla"]); abline(0, 1)
```




```
cor(df_sim[genitor_names, "Y"], df_sim[genitor_names, "Y_pred_inla"])^2
```

```
## [1] 0.3604146
```

Prédiction des BLUPs des géniteurs à partir des équations d'Anderson

```
df_sim_desc$Individual = as.character(df_sim_desc$id)
df_sim_desc = left_join(df_sim_desc, map)
```

```
library(INLA)

vtot <- var(df_sim_desc$Y, na.rm=TRUE)
s2max <- 10*vtot
prec.Random <- list(prior="pc.prec", param=c(s2max,0.01), fixed=FALSE)
prec.A <- list(prior="pc.prec", param=c(s2max, 0.01), fixed=FALSE)
```

```
fit_inla <- inla(Y ~ 1 +
                   f(newId, model="generic0", Cmatrix=Ainv_pedigreemm, constr=FALSE, hyper=list(theta=prec.
                 data=df_sim_desc,
                 family="gaussian",
                 control.family=list(hyper=list(theta=prec.Random)),
                 control.compute=list(dic=TRUE)
)

summary(fit_inla)
```

```
## Time used:
##     Pre = 2.48, Running = 2.44, Post = 0.354, Total = 5.27
## Fixed effects:
##               mean    sd 0.025quant 0.5quant 0.975quant  mode kld
## (Intercept) 9.286 0.913      7.481    9.288     11.083 9.288   0
##
## Random effects:
##   Name     Model
##     newId Generic0 model
##
## Model hyperparameters:
##                                          mean    sd 0.025quant 0.5quant
## Precision for the Gaussian observations 0.420 0.075      0.293    0.413
## Precision for newId                     0.234 0.067      0.129    0.226
##                                         0.975quant  mode
## Precision for the Gaussian observations      0.585 0.400
## Precision for newId                          0.392 0.209
##
## Deviance Information Criterion (DIC) ...............: 4975.89
## Deviance Information Criterion (DIC, saturated) ....: 1690.06
## Effective number of parameters .....................: 461.23
##
## Marginal log-Likelihood:  -3330.98
##  is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```
## les variances résiduelle et génétique
1/fit_inla$summary.hyperpar$mean
```

```
## [1] 2.381708 4.264465
```

```
head(fit_inla$summary.random$newId)
```

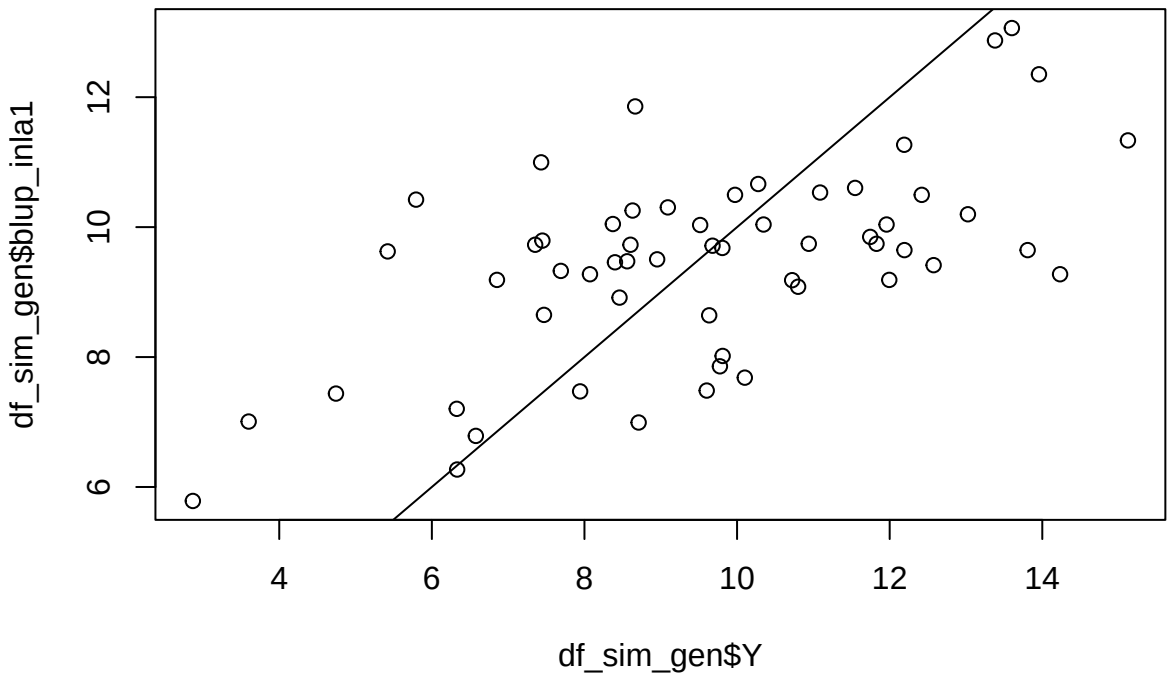
```
##   ID       mean       sd 0.025quant   0.5quant 0.975quant       mode
## 1  1 -0.1423120 2.121283  -4.330481 -0.1419175   4.043752 -0.1418981
## 2  2 -0.1423120 2.121283  -4.330481 -0.1419175   4.043752 -0.1418981
## 3  3  0.8382736 2.127904  -3.361445  0.8380933   5.038857  0.8380628
## 4  4  0.3116594 2.113127  -3.860917  0.3122641   4.480894  0.3122684
## 5  5 -0.2292300 2.000148  -4.178466 -0.2286808   3.716899 -0.2286929
## 6  6 -0.2292300 2.000148  -4.178466 -0.2286808   3.716899 -0.2286929
##            kld
```

```
## 1 5.414897e-08
## 2 5.414897e-08
## 3 5.405918e-08
## 4 5.387093e-08
## 5 5.326625e-08
## 6 5.326625e-08
```

```
## R2 descendant
cor(fit_inla$summary.fitted.values$mean, df_sim_desc$Y)^2
```

```
## [1] 0.8186179
```

```
mu_est = 9.417
all_blup_inla = fit_inla$summary.random$newId[, 1:2]
blup_inla_desc = all_blup_inla$mean[df_sim_desc$newId]

df_sim_gen$blup_inla1 = mu_est + as.vector(A21 %*% solve(A11) %*% blup_inla_desc)


plot(df_sim_gen$Y, df_sim_gen$blup_inla1); abline(0, 1)
```




```
corinla = cor(df_sim_gen$Y, df_sim_gen$blup_inla1)^2
corinla
```

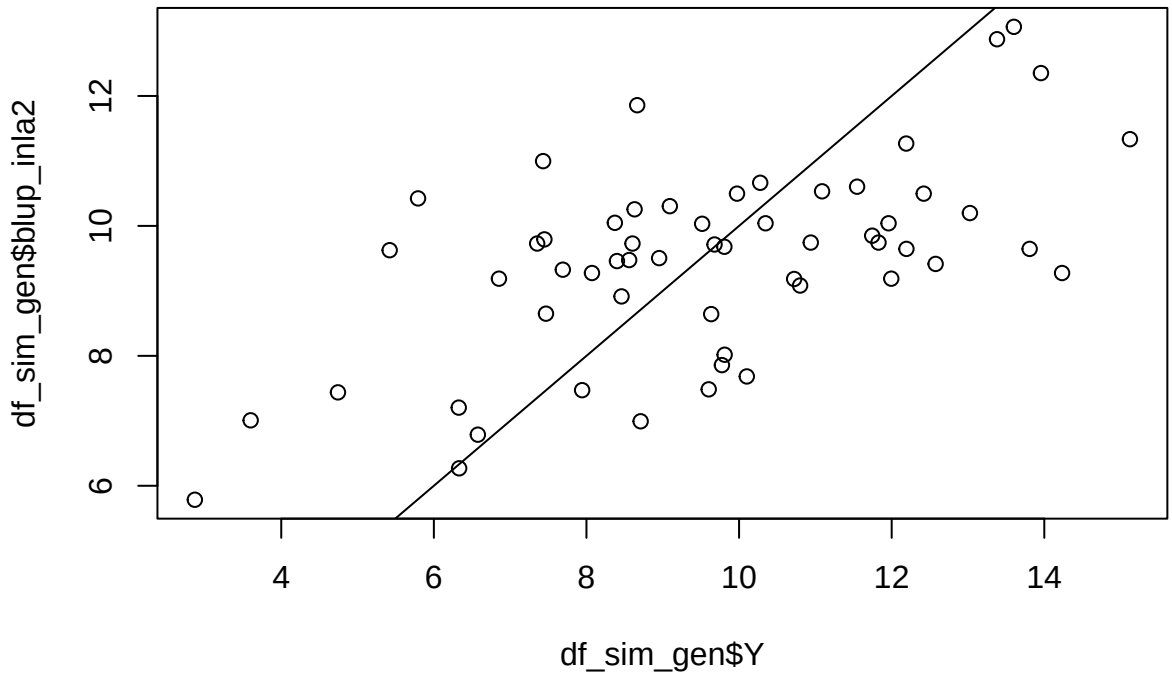
```
## [1] 0.3604663
```

On peut aussi récupérer les BLUPs des géniteurs car on a utilisé la matrice A tout entière. INLA calcule ainsi les BLUPs pour tous les niveaux, pas uniquement ceux présent dans le jeu de données (`df_sim_desc` ici)

```
df_sim_gen$Individual = as.character(df_sim_gen$id)
df_sim_gen = left_join(df_sim_gen, map)
df_sim_gen$blup_inla2 = mu_est + all_blup_inla$mean[df_sim_gen$newId]
cor(df_sim_gen$blup_inla1, df_sim_gen$blup_inla2)
```

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

```
plot(df_sim_gen$Y, df_sim_gen$blup_inla2); abline(0, 1)
```




```
corinla = cor(df_sim_gen$Y, df_sim_gen$blup_inla2)^2
corinla
```

```
## [1] 0.3604642
```

5.2.7 AsReml

Prédiction des BLUPs des géniteurs à partir des équations d'Anderson

```r
library(asreml)
ai <- asreml::ainverse(ped1)
df_sim_desc$id = as.factor(df_sim_desc$id)
fit_asreml <- asreml(Y ~ 1, random = ~vm(id, ai), data = df_sim_desc)
detach("package:asreml", unload = TRUE)

summary(fit_asreml)$varcomp
```

```r
## R2 descendants
fitted_asreml = fit_asreml$linear.predictors
plot(df_sim_desc$Y, fitted_asreml); abline(0, 1)
cor(df_sim_desc$Y, fitted_asreml)^2

## BLUP descendant
all_blup_asreml = fit_asreml$coefficients$random
blup_desc = all_blup_asreml[paste0("vm(id, ai)_", descendants_names), ]

mu_est = fit_asreml$coefficients$fixed[1, 1]
blup_gen = mu_est + as.vector(A21 %*% solve(A11) %*% blup_desc)


plot(df_sim_gen$Y, blup_gen); abline(0, 1)
corinla = cor(df_sim_gen$Y, blup_gen)^2
corinla
```

Comme pour INLA, on peut récupérer les BLUPs des géniteurs directement dans les sorties du modèle car on a utilisé la matrice A tout entière. AsReml calcule ainsi les BLUPs pour tous les niveaux, pas uniquement ceux présent dans le jeu de données (`df_sim_desc` ici)

```r
## BLUP géniteurs
blup_gen2 = mu_est + all_blup_asreml[paste0("vm(id, ai)_", genitor_names), ]
plot(df_sim_gen$Y, blup_gen2); abline(0, 1)
cor(df_sim_gen$Y, blup_gen2)^2
```

Globalement, toutes les librairies semblent donner des résultats très proches. L'estimation des BLUPs par l'équation d'Anderson donne de meilleurs résultats comparé à l'imputation des données manquantes dans la variable réponse.