**TP1 Régression bayésienne**

**ENSAI - 2022-2023**

L'objectif de ce TP est de comparer le modèle de régression aléatoire (Random Régression), la régression bayésienne A, la régression bayésienne LASSO et la méthode de sélection bayésienne SSVS. Nous étudierons le modèle linéaire suivant :

$$Y \quad = \quad \mu \mathbb{I} + X\beta + \epsilon$$

Nous utiliserons des simulations pour comparer les différentes méthodes. Nous générons un jeu de données de 200 observations. Nous découperons en deux l'échantillon pour avoir un jeu «d'apprentissage » qui servira à construire les modèles et un jeu « test » qui servira à comparer nos modèles. On coupera ainsi l'échantillon en deux : 100 observations pour l'apprentissage et 100 pour le test. On génère 300 variables explicatives i.i.d., chacune suivant une loi uniforme entre -5 et 5 (on ne standardisera pas car i.i.d.).

```
simuvarexpl <- matrix(rep(0,300*200),ncol=300,nrow=200)
for (i in 1 :300) simuvarexpl[,i] <- runif(200,-5,5)
simuvarexpl <- as.data.frame(simuvarexpl)
```

Pour générer un vecteur d'observation $Y$ nous utilisons uniquement 5 des 300 variables générées : celles d'indices 10, 20, 30, 40 et 50, que l'on associe au vecteur de paramètres de régression $\beta = (1, -1, 2, -2, 3)$. Nous supposons que $\mu = 0$.

```
trueind <- c(10,20,30,40,50)
beta <- c(1,-1,2,-2,3)
ysimu <- as.matrix(simuvarexpl)[,trueind]%*% beta + rnorm(200,0,2)
```

# 1 Fonctions utiles

## 1.1 Prédictions

La fonction suivante sera utilisée pour prédire une variable d'intérêt à partir de variables explicatives et à partir d'estimateurs de $\mu$ et $\beta$ (dans le modèle linéaire).

```
predictions <- function(matableTest,muchap,betachap){
ychap <- muchap * rep(1,dim(matableTest)[1])+ as.matrix(matableTest[,]) %*% betachap
return(ychap)}
```

## 1.2 Sélections

La fonction suivante sera utilisée en sélection de variables. Elle donne un sous-ensemble de variables pertinentes à sélectionner à partir de :

- **ResAlgo** : résultat d'un algorithme bayésien (soit les paramètres $\beta$ de régression associés aux variables explicatives disponibles, ou bien les nombres de sélection post-burn-in des variables explicatives disponibles dans le cas de SSVS).
- **varexpl** : table donnant les variables explicatives observées.
- **mini** : seuil minimal en-dessous duquel on souhaite que les variables soient sélectionées.

— **maxi** : seuil maximal au-dessus duquel on souhaite que les variables soient sélectionnées.

```
subsetSelected <- function(resAlgo, varexpl, mini, maxi) {
numselected <- c(which(resAlgo < mini), which(resAlgo > maxi))
selected <- character()
valeurs <- numeric()
for (i in 1 :length(numselected)) {
selected[i] <- names(varexpl)[numselected[i]]
valeurs[i] <- resAlgo[numselected[i]] }
subset <- cbind(selected, valeurs)
subset <- as.data.frame(subset)
subset$valeurs <- as.numeric(as.vector(subset$valeurs))
return(subset) }
```

# 2 RR-BLUP

## 2.1 Estimation

A l'aide de la fonction *mixed.solve* du package *rrBLUP* on estime les paramètres $\beta$ et $\mu$ du modèle aléatoire.

```
library(rrBLUP)
resBLUP <- mixed.solve(ysimu[1 :100], Z = as.matrix(simuvarexpl[1 :100, ]), X = as.matrix(rep(1, 100)), method = "REML")
muchap <- resBLUP$beta
betachap <- resBLUP$u
resBLUP$Ve
resBLUP$Vu
```

On s'aperçoit qu'il y a une forte variabilité dans l'estimation de la variance résiduelle...

## 2.2 Prédiction

A l'aide des estimations obtenues, nous pouvons obtenir des prédictions pour les observations du jeu de validation (observations 101 à 200 du jeu de données), et comparer ces prédictions aux valeurs observées.

```
predBLUP <- predictions(simuvarexpl[101 :200, ], muchap, betachap)
cor(ysimu[101 :200], predBLUP)
plot(predBLUP ~ ysimu[101 :200])
```

## 2.3 Sélection

A l'aide des estimations obtenues, on peut sélectionner les variables explicatives qui paraissent les plus pertinentes.

```
plot(betachap,xlab="Indice variable",ylab="Paramètre de régression associé",main="rrBLUP")
bb <- boxplot(betachap)
```

```
varselectedBLUP <- subsetSelected(betachap,simuvarexpl,bb$stats[1,], bb$stats[5,])
varselectedBLUP
```

# 3 Bayes A

## 3.1 La fonction

La fonction suivante code le Gibbs sampler associé à Bayes A. Les arguments sont $y$ le vecteur des observations, $X$ la matrice des variables explicatives disponibles, *a, b, c, et d* les hyper-paramètres des a priori inverse-gamma associés à la variance des paramètres de régression et à la variance du terme d'erreur, *muinit* est la valeur initiale spécifiée par l'utilisateur pour l'intercept du modèle, *nbiter* le nombre d'itérations total, et *nburn* le nombre d'itérations burn-in. En examinant cette fonction on retrouve les différentes étapes du Gibbs sampler.

```
library(MCMCpack) ; library(LearnBayes) ; library(mvtnorm) ; library(mnormt)
BayesA <- function(y,X,a,b,c,d,muinit,nbiter,nburn){
p <- dim(X)[2]
n <- dim(X)[1]

# resultats a garder
resbeta <- matrix(rep(0,p*(nbiter-nburn)),nrow=p,
ncol=(nbiter-nburn))
ressigma2beta <- matrix(rep(0,p*(nbiter-nburn)),nrow=p,
ncol=(nbiter-nburn))
resmu <- rep(0,nbiter-nburn)
ressigma2eps <- rep(0,nbiter-nburn)

# initialisation
beta <- rep(0,p)
mu <- muinit
sigma2beta <- rinvgamma(p,a,b)
sigma2eps <- rinvgamma(1,c,d)

#iterations
for (iter in 1 :nbiter){
print(iter)
Sigmabeta <- solve(t(X)%*%X/sigma2eps + diag(1/sigma2beta))
beta <- as.numeric(rmnorm(1,Sigmabeta%*%t(X)%*%(y-mu*rep(1,n))/sigma2eps,
Sigmabeta))
mu <- rnorm(1,t(rep(1,n))%*%(y-X%*%beta)/n,sqrt(sigma2eps/n))
for (j in 1 :p){
sigma2beta[j] <- rinvgamma(1,a+1/2,b+1/2*beta[j]^2)
}
sigma2eps <- rinvgamma(1,c+n/2,d+1/2*t(y-mu*rep(1,n)-X%*%beta)%*%
(y-mu*rep(1,n)-X%*%beta))
if (iter > nburn){
resbeta[,iter-nburn] <- beta
ressigma2beta[,iter-nburn] <- sigma2beta
```

```
resmu[iter-nburn] <- mu
ressigma2eps[iter-nburn] <- sigma2eps } }
return(list(resbeta,ressigma2beta,resmu,ressigma2eps)) }
```

### 3.2 Estimation

A l'aide de la fonction *BayesA* on peut donner des estimations des paramètres du modèle. On peut aussi examiner les traces de ces paramètres.

```
priora <- 2
priorb <- 1
priorc <- 2
priord <- 1
resBAYESA <- BayesA(ysimu[1 :100], as.matrix(simuvarexpl[1 :100, ]), priora, priorb, priorc,
priord, mean(ysimu[1 :100]), 1000, 500)
moybeta <- apply(resBAYESA[[1]], 1, mean)
moysigma2beta <- apply(resBAYESA[[2]], 1, mean)
moymu <- mean(resBAYESA[[3]])
moysigma2eps <- mean(resBAYESA[[4]])
plot(resBAYESA[[1]][1, ])
plot(resBAYESA[[2]][1, ])
plot(resBAYESA[[3]])
plot(resBAYESA[[4]])
```

On peut comparer la distribution a posteriori obtenue pour $\sigma^2_{\beta_1}$ avec sa distribution a priori.

```
tracesigmabeta1 <- resBAYESA[[2]][1,]
plot(density(tracesigmabeta1),col="blue",lwd=2, main="sigma_beta_1 : Bayes A")
curve(dinvgamma(x,2,1), add=TRUE, col="red",lwd=2,lty=4)
lalegende=c("Posterior","Prior")
legend("topright",lalegende,lwd=c(2,2),lty=c(1,4),col=c("blue","red"))
```

De même on peut comparer la distribution a posteriori obtenue pour $\sigma^2_{\beta_50}$ avec sa distribution a priori.

```
tracesigmabeta50 <- resBAYESA[[2]][50,]
plot(density(tracesigmabeta50),col="blue",lwd=2, main="sigma_beta_50 : Bayes A")
curve(dinvgamma(x,2,1), add=TRUE, col="red",lwd=2,lty=4)
lalegende=c("Posterior","Prior")
legend("topright",lalegende,lwd=c(2,2),lty=c(1,4),col=c("blue","red"))
```

On peut ici faire varier les hyperparamètres pour voir leurs influences...

### 3.3 Prédiction

A l'aide des estimations obtenues, donner des prédictions pour les observations du jeu de validation (observations 101 à 200 du jeu de données), et comparer ces prédictions aux valeurs observées.

```
predBAYESA <- predictions(simuvarexpl[101 :200, ], moymu, moybeta)
```

```
cor(ysimu[101 :200], predBAYESA)
plot(predBAYESA ~ ysimu[101 :200])
```

### 3.4 Sélection

A l'aide des estimations obtenues, sélectionner les variables explicatives qui paraissent les plus pertinentes.

```
plot(moybeta,xlab="Indice variable",ylab="Paramètre de régression associé",main="Bayes A")
bb <- boxplot(moybeta)
varselectedBAYESA <- subsetSelected(moybeta,simuvarexpl, bb$stats[1,],bb$stats[5,])
varselectedBAYESA
```

## 4 LASSO bayésien

La fonction *BLR* du package BLR permet de faire tourner un Gibbs sampler associé au LASSO bayésien. On utilisera les paramètres suivants : $\mu, \beta_F, \sigma^2_u, u|\sigma^2_u, \beta_R, \sigma^2_{\beta_R}, \beta_L, \sigma^2_j, \lambda^2, \sigma^2_\epsilon$.

### 4.1 Estimation

On peut obtenir les estimations du modèle à partir de la fonction suivante

```
library(BLR)
LASSO_BLR <- BLR(y=ysimu[1 :100],XL=as.matrix(simuvarexpl[1 :100,]), prior=list(varE=list(df=2,S=1)
lambda=list(shape=1, rate=1, type='random',value=2)), nIter=3000,burnIn=1000, saveAt="BLR_",
thin2=1)
LASSO_BLR$mu
LASSO_BLR$varE
LASSO_BLR$bL
LASSO_BLR$tau2
LASSO_BLR$lambda
```

On peut examiner la trace de $\sigma^2_\epsilon$

```
tracevarE <- scan('BLR_varE.dat')
plot(tracevarE,type='o',xlab="iteration",ylab="varE",main="trace varE")
```

Puis on peut comparer la distribution a posteriori obtenue pour $\sigma^2_\epsilon$ avec sa distribution a priori.

```
plot(density(tracevarE),col="blue",lwd=2, main="varE : LASSO bayésien")
curve(dinvgamma(x,1,1), add=TRUE, col="red",lwd=2,lty=4)
lalegende=c("Posterior","Prior")
legend("topright",lalegende,lwd=c(2,2),lty=c(1,4),col=c("blue","red"))
```

De même on peut examiner la trace de $\lambda^2$.

```
tracelambda <- scan('BLR_lambda.dat')
plot(tracelambda,type='o',xlab="iteration",ylab="lambda",main="trace lambda")
```

Puis on peut comparer la distribution a posteriori obtenue pour $\lambda^2$ avec sa distribution a priori.

```
plot(density(tracelambda),xlim=c(0,5),col="blue",lwd=2, main="lambda : LASSO bayésien")
curve(dgamma(x,1,1), add=TRUE, col="red",lwd=2,lty=4)
lalegende=c("Posterior","Prior")
legend("topright",lalegende,lwd=c(2,2),lty=c(1,4),col=c("blue","red"))
```

On peut faire varier les hyper-paramètres pour voir leurs influences sur les a posteriori.

### 4.2 Prédiction

A l'aide des estimations obtenues, on obtient des prédictions pour les observations du jeu de validation (observations 101 à 200 du jeu de données), et l'on compare ces prédictions aux valeurs observées.

```
predLASSOBLR <- predictions(simuvarexpl[101 :200, ], LASSO_BLR$mu, LASSO_BLR$bL)
cor(ysimu[101 :200], predLASSOBLR)
plot(predLASSOBLR ~ ysimu[101 :200])
```

### 4.3 Estimation

A l'aide des estimations obtenues, on sélectionne les variables explicatives qui paraissent les plus pertinentes.

```
plot(LASSO_BLR$bL,xlab="Indice variable",ylab="Paramètre de régression associé", main="LASSO bayésien")
bb <- boxplot(LASSO_BLR$bL)
varselectedLASSO <- subsetSelected(LASSO_BLR$bL,simuvarexpl,-0.6,0.8)
varselectedLASSO
```

## 5 SSVS

La fonction *selection_SSVS* ci-dessous code l'algorithme de Metropolis Hastings associé à la méthode SSVS. Les arguments sont *vardep* le vecteur des observations (variable à expliquer), *varexpl* la table donnant les variables explicatives dont nous disposons, *nbiter* le nombre d'itérations total, *nburn* le nombre d'itérations burn-in, *lec* le facteur d'échelle c, *nbSelecInit* le nombre de variables à sélectionner initialement, *nbToChange* le nombre de variables dont le statut est modifié à chaque itération ($\gamma_j$ passe de 0 à 1, ou de 1 à 0), et *Pi* la probabilité a priori de chacune des variables d'être sélectionnée. Dans cette fonction, on peut repérer les étapes suivantes :

- Initialisation de chacun des paramètres.
- Proposition d'un nouveau vecteur.
- Calcul de la probabilité de Metropolis-Hastings.
- Etape de Metropolis-Hastings.
- Les différents résultats retournés par la fonction.

```
selection_SSVS <- function (vardep, varexpl, nbiter, nburn, lec, nbSelecInit,nbToChange,Pi)
{
```

```
X <- as.matrix(varexpl)
y <- as.numeric(as.vector(vardep))
y <- y - mean(y)
nind <- dim(X)[1]
nvar <- dim(X)[2]
sumgamma <- rep(0, nvar)
nbactu <- 0
nbselecactu <- numeric()
indgamma10 <- sample(c(1 :nvar), nbSelecInit, replace = FALSE)
gamma0 <- rep(0, nvar)
for (i in 1 :nbSelecInit) {
gamma0[indgamma10[i]] <- 1 }
indgamma1 <- indgamma10
gamma <- gamma0
nbSelec <- nbSelecInit
for (iter in 1 :nbiter) {
print(iter)
gammaprop <- gamma
indgamma1prop <- indgamma1
indToChange <- sample(c(1 :nvar), nbToChange, replace = FALSE)
for (i in 1 :nbToChange){
if (gamma[indToChange[i]]==0){
gammaprop[indToChange[i]] <- 1
indgamma1prop <- c(indgamma1prop,indToChange[i])
}
else {
gammaprop[indToChange[i]] <- 0
indremove <- which(indgamma1prop==indToChange[i])
indgamma1prop <- indgamma1prop[-indremove]
} }
nbSelecprop <- length(indgamma1prop)
if (nbSelecprop==0){ # condition pour empecher gamma avec que des 0
cond <- 0
while(cond==0){
gammaprop <- gamma
indgamma1prop <- indgamma1
indToChange <- sample(c(1 :nvar), nbToChange, replace = FALSE)
for (i in 1 :nbToChange){
if (gamma[indToChange[i]]==0){
gammaprop[indToChange[i]] <- 1
indgamma1prop <- c(indgamma1prop,indToChange[i])
}
else {
gammaprop[indToChange[i]] <- 0
indremove <- which(indgamma1prop==indToChange[i])
indgamma1prop <- indgamma1prop[-indremove]
} }
nbSelecprop <- length(indgamma1prop)
if (nbSelecprop>0){cond <- 1}
```

```
} }
indgamma1 <- which(gamma == 1)
nbSelec <- length(indgamma1)
Xgamma <- X[, indgamma1]
Xgammaprop <- X[, indgamma1prop]
temp <- (t(y)%*%(diag(rep(1,nind))-lec/(1+lec)*Xgammaprop
%*%solve(t(Xgammaprop)%*%Xgammaprop)%*%t(Xgammaprop))%*%y)/
(t(y)%*%(diag(rep(1,nind))-lec/(1+lec)*Xgamma
%*%solve(t(Xgamma)%*%Xgamma)%*%t(Xgamma))%*%y)
A <- (1+lec)^((nbSelec-nbSelecprop)/2)*(Pi/(1-Pi))^(nbSelecprop-nbSelec)* temp^(-(nind-1)/2)
probaccept1 <- min(1,A)
seuil <- runif(1)
if (seuil < probaccept1){
gamma <- gammaprop
indgamma1 <- indgamma1prop
nbSelec <- nbSelecprop
nbactu <- nbactu+1
nbselecactu <- c(nbselecactu,nbSelec)
}
if (iter > nburn) {
sumgamma <- sumgamma + gamma
} }
return(list(sumgamma,nbactu,nbselecactu))
}
```

Cette fonction renvoie

- *sumgamma* un vecteur de taille p dont le jème élément donne le nombre de fois lors des itérations post-burn-in où la jème variable a été sélectionnée dans le modèle
- *nbactu* qui indique le nombre de fois où des changements ont été acepté.
- *nbselecatcu* qui désigne le nombre de variable sélectionnées lors des *nbactu* changements.

Comme on s'intéresse qu'à l'estimation du vecteur indicateur $\gamma$ ici, il n'y a pas d'a priori ou d'hyperparamètres à régler. Seulement la probabilité $\pi$ de choisir un candidat. Et le coefficient de sélection, noté *lec*.

On peut faire tourner la fonction *selection_SSVS* sur les simulations et examiner les résultats obtenus.

```
nbinit <- 10
nbToChange <- 2
Pi <- 10/300
lec <- 50
resSSVS <- selection_SSVS(ysimu[1 :100], as.matrix(simuvarexpl[1 :100, ]), 3000, 1000, lec,
nbinit, nbToChange, Pi)
resSSVS[[2]]
resSSVS[[3]]
resSSVS[[1]]
```

C'est le vecteur resSSVS[[1]] qui va nous intéresser car il contient les nombres de fois où les variables ont été sélectionnées (correspondant aux coefficients $\beta_j$ non nuls).

A l'aide des résultats obtenus on peut sélectionner les variables explicatives qui paraissent les plus pertinentes.

```
boxplot(resSSVS[[1]])
plot(resSSVS[[1]],xlab="Indice variable",ylab= "Nombre de sélections post-burn-in", main="SSVS")
varselectedSSVS <- subsetSelected(resSSVS[[1]],simuvarexpl,0,1500)
varselectedSSVS
```