TP0 Régression bayésienne

ENSAI - 2024/2025

L'objectif de ce TP est de s'initier à la programmation bayésienne. Nous étudierons le modèle linéaire suivant :

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

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 μ et β (dans le modèle linéaire).

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

2 Simulation d'un coefficient aléatoire

On suppose ici que $\mu = 0$ (modèle sans constante). On considère les lois suivantes :

$$\beta \sim \mathcal{N}(0, G) \qquad \epsilon \sim \mathcal{N}(0, R)$$

avec $G = \sigma_\beta^2 I$ et $R = \sigma_\epsilon^2 I$ (I désigne l'identité).

- Calculer la loi de $Y|\beta$.
- Calculer la loi a posteriori de $\beta|Y$.
- Simuler par Gibbs sampler la chaîne de Markov associée à $\beta|Y$.
- Sensibilité : étudier l'impact de la valeur de σ_β^2 sur l'estimation de $\beta|Y$.
- Représentez des trajectoires de "vrais" coefficients et de "faux" coefficients.

3 Ajout d'un a priori de Zellner

On suppose ici que $\mu = 0$ (modèle sans constante). On considère les lois suivantes :

$$\beta \sim \mathcal{N}(0, \sigma^2(X'X)^{-1}) \qquad \epsilon \sim \mathcal{N}(0, R)$$

avec $R = \sigma_\epsilon^2 I$. En général on choisit $\sigma^2 = c\sigma_\epsilon^2$. Proposez une version ridge de la loi de β pour pouvoir simuler la loi a posteriori de $\beta|Y$.

4 Ajout d'une loi sur σ_ϵ^2

On suppose ici que $\mu = 0$ (modèle sans constante). L'inconvénient des modèles précédents est de ne pouvoir estimer la variance résiduelle σ_ϵ^2. On peut soit faire des estimations RR-BLUP. Soit continuer en bayésien. Ici on ajoute une loi (a priori)sur σ_ϵ

$$\beta \sim \mathcal{N}(0, R) \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2 I) \quad \sigma_\epsilon^2 \sim \mathcal{IG}(c, d)$$

avec c, d les hyperparamètres de la loi Inverse Gamma. Simulez une chaîne de Markov par Gibbs sampler pour estimer les paramètres du modèle.

5 Ajout d'une loi sur σ_β^2

L'a priori sur β a une forte influence. On peut ajouter des hyperparamètres et s'approcher du Bayes A.

$$\begin{aligned} \beta \sim \mathcal{N}(0, diag(\sigma_{\beta_1}^2, \cdots, \sigma_{\beta_p}^2) \qquad & \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2 I) \\ \sigma_{\beta_j}^2 \sim \mathcal{IG}(a, b) \qquad & \sigma_\epsilon^2 \sim \mathcal{IG}(c, d) \end{aligned}$$

avec a, b, c, d les hyperparamètres de la loi Inverse Gamma. On prend ici des σ_{β_j} différents pour laisser chaque coefficients varier librement. On peut ajouter également une moyenne μ et une loi vague, par exemple uniforme.

6 Metropolis-Hasting

On peut reprendre les exercices précédents en utilisant un algorithme de Metropolis-Hasting.

---

7 Exemple de simulation par ABC

On considère des observations gaussiennes $Y_1, \cdots, Y_n$ qui suivent la même loi normale $\mathcal{N}(\mu, \sigma^2)$, avec $n = 100$, $\mu = 1$ et $\sigma^2 = 4$.

On veut estimer la moyenne μ et la variance σ^2 par la méthode ABC.

Nous n'avons pas d'a priori précis sur μ et σ et nous mettons par ex. des lois uniformes (dont on peut faire varier les hyperparamètres) :

simMU <- function () {

return (runif(1, min=-10, max=10)) }
simSIGMA = function () {
return (runif(1, min=0, max=10)) }

On simule des $Y_1^*, \cdots, Y_n^*$ candidats à ressembler aux observations :

simul = function (n, mu, sigma) {
return(rnorm(n, mean = mu, sd = sigma)) }

Il faut maintenant comparer les simulations aux observations et accepter ou non l'égalité. Pour cela il faut résumer les données et comparer les "summary statistics". On peut par exemple

- Faire un test d'égalité des lois des données.
- Comparer des fractiles des données.
- Compare les moyennes et les variances des données.

Distance basée sur la moyenne et la variance
MeanVar = function (obs, simu) {
diffmean <- abs(mean(obs) - mean(simu))
diffsd <- abs(sd(obs) - sd(simu))
return(c(diffmean, diffsd)) }
La fonction d'acceptation-rejet va être de la forme (1 si accepté, 0 sinon) :
acceptMeanVar = function (obs, simu, seuil) {
distances = MeanVar(obs, simu)
if((differences[1] < seuil[1]) & (differences[2] < seuil[2])) return(1) else return(0) }

Distance basée sur les quantiles
quant = function(data) {
return (quantile(data, probs=c(0.1, 0.5, 0.9))) }
comparequant = function (obs, simu) {
*compare = sqrt(sum(mapply(function(x,y) (x-y)**2, obs, simu)))*
return(compare) }
La fonction d'acceptation-rejet va être de la forme (1 si accepté, 0 sinon) :
acceptQuant = function (obs, simu, seuil) {
distances = comparequant(quant(obs), quant(simu))
if((distances[1] < seuil[1]) & (distances[2] < seuil[2]))return(1) else return(0)

Programme global
sampled = function (obs, nIter, seuil, acceptfunction) {
n = length(obs)
decision = vector(length = nIter)
sampledMU = vector(length = nIter, mode = "numeric")
sampledSIGMA = vector (length = nIter, mode = "numeric")
for (i in 1 :nIter){
mu <- simMU()
sigma <- simSIGMA()
simuDATA = simu(n, mu, sigma)
decision[i] = acceptfunction(obs, simuDATA, seuil)
sampledMU[i] = mu

sampledSIGMA[i] = sigma }
return(data.frame(cbind("DECISION" = decision, "sampledMU" = sampledMU, "sampled-SIGMA" = sampledSIGMA))) }

Mise en pratique
resu = sampled(data, n, seuil, accept)
Pourcentage d'acceptation :
sum(resu$decision)/n
Indices acceptés :
ind=which(resu$DECISION==1)
Trace et densités des paramètres :
resMU= res$sampledMU
*resSIGMA2=res$sampledSIGM**2*
plot(density(resMU))
mmean(resMU)
plot(density(resSIGMA2))
mean(resSIGMA)

Extensions On pourrait considérer un modèle de régression avec $Y|\beta, \sigma^2 \sim \mathcal{N}(X\beta, \sigma^2 I)$.