TP 4

Partie I : B.A.-ba de l'ABC

L'objectif de cette partie est de mettre en place une fois l'algorithme ABC.

1. On considère (encore !) un modèle
$$X_i \mid p \sim_{i.i.d.} \mathcal{B}er(p),$$
et on prend un prior sur p donné par
$$p \sim Beta\Big(\frac{1}{2}, \frac{1}{2}\Big)$$
   (a) Pour l'instant, on n'utilise pas la propriété de conjugaison du prior. En simulant le prior, et en utilisant une méthode du rejet, simuler la loi à posteriori $\pi(p \mid X_1 = 1)$, après une observation ayant donnée 1. On écrira pour cela une fonction `simul_1` qui prend en argument les hyperparamètres du prior a, b et x_1, et qui simule le posterior $p \mid X_1 = x_1$, en ayant choisi un prior $Beta(a, b)$ pour le prior.
   (b) En reproduisant de nombreuses simulations, comparer la loi de la variable simulée à la question précédente, avec la loi théorique (obtenue par propriété de conjugaison du prior).
2. On voudrait faire de même pour simuler la loi de $p \mid (X_1, \cdots X_{20}) = (1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1)$. Quel problème commence-t-on à rencontrer. Que va t-il se passer pour des observations beaucoup plus nombreuses ?
3. On pose $S_n = \sum_{i=1}^n X_i$. Quelle propriété que vérifie S_n permet d'écrire
$$\pi(p \mid X_1, \cdots X_n) = \pi(p \mid S_n)?$$
4. Comment simuleriez-vous la loi de $p \mid S_n$ en utilisant la même idée que précédemment ? On écrira pour cela une fonction `simul_2` qui prend en argument les hyperparamètres du prior a, b, n et s, et qui simule le posterior $p \mid S_n = s$, en ayant choisi un prior $Beta(a, b)$ pour le prior.
5. Écrire une fonction `rsimABC` qui simule sous la loi
$$p \mid |S_n - s| < \epsilon.$$
6. Utiliser la fonction précédente pour simuler sous une approximation de la loi a posteriori, pour un échantillon de 1000 observations tel que $S_n = 750$ et comparer à la loi théorique. (on pourra prendre $\epsilon = 30$ par exemple).

Partie II : Metropolis-Hasting : Modèle de mélange

L'objectif de cette partie est de traiter le cas d'un mélange Gaussien (à deux composantes pour le moment).

$$f_{X|p,m,\tau}(x) = p f_{\mathcal{N}(m_1, \frac{1}{\tau_1})}(x) + (1-p) f_{\mathcal{N}(m_2, \frac{1}{\tau_2})}(x),$$

où l'on note $m = (m_1, m_2)$, et $\tau = (\tau_1, \tau_2)$.)

1. Écrire une fonction `rmix` qui prend en argument la taille `n` de l'échantillon que l'on souhaite simuler, ainsi qu'une liste `l` de paramètres `l$m,l$tau,l$p`, et qui renvoie un vecteur de réalisations du mélange correspondant à ces paramètres. Représenter un histogramme des données simulées, avec les paramètres $p = 0.6, m = (0, 5)$ et $\tau = (1, 1)$
2. Écrire une fonction `logvraiss` qui prend en argument un vecteur d'observations `x` et une liste `l` de paramètres `l$m,l$tau,l$p` et qui renvoie la log-vraissemblance de l'échantillon, correspondant à ces paramètres.

3. On choisit les priors suivants :

$$\begin{aligned} p &\sim Beta(a_0, b_0) \\ m_i &\sim \mathcal{N}\left(\mu_0, \sigma_0^2\right), i = 1, 2 \\ \tau_i &\sim Gamma(k_0, \lambda_0), i = 1, 2 \\ X_i | p, m, \tau &\sim_{i.i.d} \end{aligned}$$

, pour des hyperparamètres $a_0, b_0, \mu_0, \sigma_0, k_0, \lambda_0$ que l'on fixera ultérieurement. Ècrire une fonction `logprior` qui prend en argument la liste `l` de paramètres, ainsi qu'une liste `hyperparam` d'hyperparamètres, et qui renvoie le log de la densité du prior.

4. Ècrire une fonction `logpost` qui prend en argument un vecteur d'observations `x`, la liste `l` de paramètres, ainsi qu'une liste `hyperparam` d'hyperparamètres, et qui renvoie le log de la densité du posterior, à une constante additive (qui dépend des données) près.

5. Il reste à choisir une loi de proposition. On propose le code suivant, à adapter éventuellement.

```
rprop = function(l,lsig =list(p = 1/100,m1 = 1/10,m2 = 1/10,tau1 = 1/100,tau2 = 1/100)){
 logratio = 0
 p = rbeta(1,1+l$p/lsig$p,1+(1-l$p)/lsig$p)
 logratio = logratio+log(dbeta(l$p,1+p/lsig$p,1+(1-p)/lsig$p))-log(dbeta(p,1+l$p/lsig$p,1+(1-l$p)/lsig$p))
 m1 = rnorm(1,l$m1,lsig$m1)
 m2 = rnorm(1,l$m2,lsig$m2)
 tau1 = rgamma(1,l$tau1/lsig$tau1,1/lsig$tau1)
 logratio = logratio+log(dgamma(l$tau1,tau1/lsig$tau1,1/lsig$tau1))-log(dgamma(tau1,l$tau1/lsig$tau1,1/lsig$tau1
 tau2 = rgamma(1,l$tau2/lsig$tau2,1/lsig$tau2)
 logratio = logratio+log(dgamma(l$tau2,tau2/lsig$tau2,1/lsig$tau2))-log(dgamma(tau2,l$tau2/lsig$tau2,1/lsig$tau2
 if (m1>m2){##pour l'identifiabilié, on demande en plus à ce que m_1<m_2
   provm = m1
   provtau = tau1
   tau1 = tau2
   m1 = m2
   m2 = provm
   tau2 = provtau
   p = 1-p
 }
 param = list(p = p,m1 = m1,m2 = m2,tau1 = tau1,tau2 = tau2)
 return(list(param = param,logratio = logratio))
}
```

Expliquer ce que fait le code fourni, et ajoutez des commentaires pertinents. N'hésitez pas à l'adapter, à la modifier ou à la traduire...

6. Écrire une fonction `stepMH` qui prend en argument les données `x`, la liste de paramètres de l'étape actuelle de l'algorithme de Metropolis Hasting `l`, et la liste d'hyperparamètres `hyperparam` et qui renvoie les paramètres de l'étape suivante, ainsi qu'un booléen vallant `T` si la proposition a été acceptée.

7. On propose l'initialisation suivante des paramètres, expliquez ce choix

```
init = function(x){
  tau1 = tau2 =1/var(x)
  m1 = m2 = mean(x)
  p =1/2
  return(list(p = p,m1 = m1,m2 = m2,tau1 = tau1,tau2 = tau2))
}
```

8. Écrire la fonction `run_MH` qui prend en arguments les données `x`, et un nombre d'étapes `nbstep`, et qui renvoie un tableau contenant pour chaque paramètre, la suite des valeurs obtenues à chaque étape de l'algorithme, ainsi que le vecteur de booléens disant si les propositions ont été acceptées.

9. Représenter les trajectoires pour différents paramètres. Qu'observe-t-on au début de l'algorithme ? Quel est la proportion de rejet ? Que se passe-t-il si l'on modifie la variance des lois de propositions ?
10. Représenter la loi a posteriori pour différents paramètres, calculer l'espérance et la variance à posteriori de chaque paramètre, et construire des régions de crédibilité pour (m_1, m_2).
11. Utiliser l'algorithme précédemment développé sur de vraies données, de votre projet, ou d'`Howell1_censured.csv` qui recense la taille, le poids, l'âge et le sexe de plusieurs individus.[1]
12. Apprendre à utiliser le package `rjags` pour simuler sous la loi a posteriori. On pourra s'inspirer de l'exercice 20 du polycopié.
13. Reprendre le même exercice en explicitant la variable de mélange, et en utilisant l'algorithme de Gibbs. En déduire pour chaque X_i la probabilité a posteriori d'appartenance à chaque classe du mélange.
14. Comparer les résultats obtenus avec un algorithme EM.

[1] "The data contained in data (Howell1) are partial census data for the Dobe area !Kung San, compiled from interviews conducted by Nancy Howell in the late 1960s."