Projet STA211 - Ajustement d'une loi de Weibull sur des données de durée de vie d'un composant industriel, avec censures à droite

Blandine Chabert et Marine Berthier

On cherche à estimer la distribution $\mathcal{P}$ de la durée de vie T d'un composant industriel (batterie de portable ou de voiture, turbine d'une centrale à énergie renouvelable, ...). On dispose pour cela d'un jeu de données de n temps de fonctionnement observés $t_1, \dots, t_n$, où :

- pour $i = 1, \dots, pt_i$ est un temps à défaillance, c'est-à-dire que la durée de vie T_i du i-ème composant est exactement t_i : $T_i = t_i$;
- pour $i = p+1, \dots, n, t_i$ est une censure à droite, c'est-à-dire que la durée de vie T_i du i-ème composant est au moins $T_i \geq t_i$;

On suppose de plus que les durées de vie suivent la loi de Weibull $\mathcal{W}_{\alpha,\kappa}$, de fonction de répartition :

$$\mathcal{P}\left[T_i \leq t \mid \alpha, \kappa\right] = F(t \mid \alpha, \kappa) = 1 - \exp\left(-\alpha t^{\kappa}\right)$$

avec $\alpha > 0$ et $\kappa > 0$.

Nous allons examiner plusieurs méthodes, fréquentistes et bayésiennes, pour estimer les paramètres α, κ à l'aide du jeu de données $t_{1:n}$, et comparer leurs résultats.

1 Simulation

- Calculer la fonction de répartition inverse $F^{-1}(x \mid \alpha, \kappa)$ pour tout $x \in [0, 1]$ de la loi de Weibull, et en déduire un algorithme de simulation de cette loi basée sur l'inversion générique.

  On résout l'équation : $F(x \mid \alpha, \kappa) = y$

  $1 - \exp(-\alpha x^{\kappa}) = y$

  $x^{\kappa} = \frac{-1}{\alpha} \log(1 - y)$

  d'où $x = F^{-1}(y \mid \alpha, \kappa) = (\frac{-1}{\alpha} \log(1 - y))^{\frac{1}{\kappa}}$

- Implémenter une fonction R qui simule n observations de la loi de Weibull de paramètres alpha et kappa donnés, censurées au-dessus d'un niveau t0 donné. Le programme renvoie le vecteur de données simulées, ordonné de telle sorte que les p premières observations ne soient pas censurées, ainsi que le nombre p.

  On fait en sorte de pouvoir facilement accéder à p en le rajoutant comme dernier élément du vecteur.

```
rweibull <- function(n, kappa=1, alpha=1, t0){
  U = runif( n )
  X = qweibull( U, kappa, alpha )
  X = sapply(X, function(x) min(t0, x))
  X=sort(X)
  p=which.max(X)
  X[n+1]=p
  return(X)
}
```

2 Calcul de la vraisemblance

- Calcul de la log-vraisemblance $\ell(t_{1:n} \mid \alpha, \kappa, p)$, dans le modèle de Weibull dont les $n-p$ dernières données sont censurées à droite

  Par définition,

$$L(t_{1:n} \mid \alpha, \kappa, p) = \prod_{i=1}^{n} f(x_i; \alpha, \kappa)$$

  On a $f(t \mid \alpha, \kappa) = \alpha\kappa t^{\kappa-1} \exp(-\alpha t^{\kappa})$

  La log-vraisemblance d'une observation non censurée ti est donnée par :

$$\ell(t_i \mid \alpha, \kappa, p) = \log(\alpha) + \log(\kappa) + (\kappa - 1)t_i - \alpha t_i^{\kappa}$$

  Pour une censure à droite, la log-vraisemblance est égale à la fonction de queue de la loi de Weibull : soit

$$\ell(t_i \mid \alpha, \kappa, p) = -\alpha t_i^{\kappa}$$

  D'où, sachant qu'il y a p observations censurées :

$$\ell(t_{1:n} \mid \alpha, \kappa, p) = p(\log\alpha + \log\kappa) + (\kappa - 1)\sum_{i=1}^{p} \log t_i - \alpha \sum_{i=1}^{n} t_i^{\kappa}$$

- Donner l'expression exacte du gradient et de la matrice hessienne de ℓ

Les éléments composants le gradient :

$$\frac{\partial \ell\left(t_{1:n} \mid \alpha, \kappa, P\right)}{\partial \alpha} = \frac{p}{\alpha} - \sum_{i=1}^{n} t_i^{\kappa}$$

$$\frac{\partial \ell}{\partial \kappa} = \frac{p}{\kappa} + \sum_{i=1}^{p} \log_i t_i - \alpha \sum_{i=1}^{n} \ln_n\left(t_i\right) t_i$$

$$\frac{\partial \ell}{\partial p} = \log \alpha + \log \kappa$$

La hessienne :

$$\mathbf{H}(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial \alpha^2} & \frac{\partial^2 f}{\partial \alpha \partial \kappa} & \frac{\partial^2 f}{\partial \alpha \partial p} \\ \frac{\partial^2 f}{\partial \kappa \partial \alpha} & \frac{\partial^2 f}{\partial \kappa^2} & \frac{\partial^2 f}{\partial \kappa \partial p} \\ \frac{\partial^2 f}{\partial p \partial \alpha} & \frac{\partial^2 f}{\partial p \partial \kappa} & \frac{\partial^2 f}{\partial p^2} \end{bmatrix}$$

$$\mathbf{H}(f) = \begin{bmatrix} \frac{-p}{\alpha^2} & -\sum_{i=1}^{n} \ln\left(t_i\right) t_i^{\kappa} & \frac{1}{\alpha} \\ -\sum_{i=1}^{n} \ln\left(t_i\right) t_i^{\kappa} & -\frac{p}{\kappa^2} - \alpha \sum_{i=1}^{n} \ln^2(ti) t_i^{\kappa} & \frac{1}{\kappa} \\ \frac{1}{\alpha} & \frac{1}{\kappa} & 0 \end{bmatrix}$$

3 Approche fréquentiste

- Déterminer le maximum de ℓ en α à κ fixé, et en déduire l'expression exacte du maximum de vraisemblance conditionnel $\widehat{\alpha}_{MLE}(\kappa) := \arg\max_{\alpha} \ell\left(t_{1:n} \mid \alpha, \kappa, p\right)$. En déduire l'expression de la vraisemblance profilée en $\kappa, \ell_{\text{prof}}\ \left(t_{1:n} \mid \kappa, p\right) := \max_{\alpha} \ell\left(t_{1:n} \mid \alpha, \kappa, p\right)$

$$\frac{\partial \ell}{\partial \alpha} = 0 \Rightarrow \frac{p}{\alpha} = \sum_{i=1}^{n} t_i^{\kappa} \Rightarrow \alpha = \frac{p}{\sum_{i=1}^{n} t_i^{\kappa}}$$

$$\widehat{\alpha}_{MLE}(\kappa) = \frac{p}{\sum_{i=1}^{n} t_i^{\hat{\kappa}}}$$

$$\ell_{prof}\left(t_{1:in} \mid \kappa, p\right) = \max_{\alpha} \ell\left(t_{1:n} \mid \alpha, \kappa, p\right)$$

En remplacant l'expression avec l'expression de l'estimateur de α :

$$\ell_{prof}\left(t_{1:in} \mid \kappa, p\right) = p(\log \frac{p}{\sum_{i=1}^{n} t_i^{\kappa}} + \log \kappa) + (\kappa - 1) \sum_{i=1}^{p} \log t_i - p$$

- À l'aide de la fonction optimize, écrire un programme R qui prend en entrée le vecteur $t = (t_1, \ldots t_n)$ des durées de fonctionnement observées et le nombre p de données non censurées, et renvoie les estimateurs du

maximum de vraisemblance de (α, κ). On supposera que $\hat{\kappa}$ prend une valeur comprise entre 0.01 et 100 .

On se sert de l'expression obtenue précédemment pour écrire le programme. On écrit la fonction de la vraisemblance profilée dépendant de κ On écrit ensuite une fonction qui cherche le maximum de la vraisemblance profilée, on obtient alors $\hat{\kappa}$ et d'après les calculs précédents on en déduit

$$\widehat{\alpha}_{MLE}(\kappa) = \frac{p}{\sum_{i=1}^{n} t_i^{\hat{\kappa}}}$$

- Utilisation du programme pour estimer (α, κ) par maximum de vraisemblance, à partir du jeu de données fourni.

  On traite le jeu de données : on le trie, on choisit $t_0 = 4$ et on trouve p. La fonction de maximisation de la vraisemblance nous donne l'estimation suivante :

```
> cat("Estimation d'alpha pour les données tests :",alpha_hat)
Estimation d'alpha pour les données tests : 0.08901503
> cat("Estimation de kappa pour les données tests :", kappa_hat)
Estimation de kappa pour les données tests : 1.820281
```

- À l'aide de la fonction écrite en première partie, simuler 10000 jeux de données $\left(t_{1:n}^{(k)}, p^{(k)}\right)$, de même taille n que le jeu de données initial, pour la même valeur seuil t_0 et en prenant (α, κ) égaux à leurs valeurs estimées sur le jeu de données fourni. Puis, pour chaque jeu de données simulé, ré-estimer :

```
n=20
alpha_re = numeric(10000)
kappa_re = numeric(10000)
q_60 = numeric(10000)
for (i in 1:10000){
  y<-rweibull(n,kappa_hat,alpha_hat,t0=4)
  max_k=max_vraisemblance_en_k(y[1:n],y[n+1])
  alpha_re[i]=max_k[1]
  kappa_re[i]=max_k[2]
  q_60[i]=qweibull(0.6,alpha_re[i],kappa_re[i])
}
```

  Grâce à l'écriture de la fonction rweibull on tient compte du fait que p varie à chaque nouvelle simulation. On obtient des échantillons dit "bootstrap" $\left(\widehat{\alpha}_{MLE}^{(k)}, \widehat{\kappa}_{MLE}^{(k)}, \widehat{q}_{60\%}\right)_{1 \leq k \leq 10000}$ représentatifs de la loi des estimateurs du maximum de vraisemblance.

- Construire des intervalles de confiance à 95% pour α, κ et $q_{60\%}$, obtenus en considérant les quantiles empiriques d'ordre 2.5% et 97.5% des échantillons bootstrap ci-dessus.

On cherche les quantiles empiriques d'ordre 2.5% et 97.5% des échantillons bootstrap.

```
alpha_IC <- quantile(alpha_re, c(0.025, 0.975))
kappa_IC <- quantile(kappa_re, c(0.025, 0.975))
q60_IC <- quantile(q_60, c(0.025, 0.975))
```

Et on obtient les résultats suivants :

```
> cat("Intervalle de confiance à 95% pour alpha : [", alpha_IC[1], ", ", alpha_IC[2], "]\n")
Intervalle de confiance à 95% pour alpha : [ 0.01077562 ,  0.2069045 ]
> cat("Intervalle de confiance à 95% pour kappa : [", kappa_IC[1], ", ", kappa_IC[2], "]\n")
Intervalle de confiance à 95% pour kappa : [ 1.274371 ,  3.413117 ]
> cat("Intervalle de confiance à 95% pour q60% : [", q60_IC[1], ", ", q60_IC[2], "]\n")
Intervalle de confiance à 95% pour q60% : [ 5.326058e-53 ,  0.1745371 ]
```

Les véritables valeurs, qui sont celles que l'on avait obtenus par première estimation sur l'échantillon test, appartiennent bien à ces deux intervalles de confiance.

4 Approche Bayésienne

Lois à postériori

- Si la loi a priori sur α est une loi Gamma $\mathcal{G}(a, b)$ alors la loi a posteriori conditionelle de α sachant κ est une loi Gamma $\mathcal{G}\left(p + a, \sum_{i=1}^{p} t_i^k + b\right)$.

Preuve:

- $\pi(\alpha) \propto \alpha^{\alpha-1} e^{-b\alpha}$
- $\pi(\alpha \mid \kappa, t_{1:n}, p) \propto L\left(t_{1:n} \mid \alpha, \kappa, p\right) \times \pi(\alpha)$
  $\propto \alpha^{a-1} e^{-b\alpha} \exp\left[p(\log\alpha + \log\kappa) + (\kappa - 1)\sum_{i=1}^{p} \log t_i - \alpha \sum_{i=1}^{n} t_i^{\kappa}\right]$

  $\propto \mathcal{G}\left(p + a, \sum_{i=1}^{p} t_i^{\kappa} + b\right)$

- Si la loi a priori sur κ est une loi Gamma $\mathcal{G}(c, d)$ alors on peut déterminier une relation de proportionalité pour la densité marginale à postériori de κ :

$$\pi\left(\kappa \mid t_{1:n}, p\right) = \frac{\pi\left(\alpha, \kappa \mid t_{1:n}\right)}{\pi\left(\alpha \mid \kappa, t_{1:n}\right)}$$

$$\propto \frac{\pi(\kappa)\pi(\alpha)\mathcal{L}\left(t_{1:n} \mid \kappa, \alpha, p\right)}{\pi\left(\alpha \mid \kappa, t_{1:n}\right)}$$

$$\propto \frac{\pi(\kappa)\alpha^{a-1}e^{-b\alpha}\alpha^{p}\kappa^{p}\prod_{i=1}^{p} t_i^{\kappa-1} e^{-\left(\sum_{i=1}^{n} t_i^{\kappa}\right)\alpha}}{\alpha^{a+p-1}e^{-\left(b+\sum_{i=1}^{n} t_i^{\kappa}\right)\alpha}\left(b + \sum_{i=1}^{n} t_i^{\kappa}\right)^{(a+p)}} = \pi(\kappa)\kappa^{p}\prod_{i=1}^{p} t_i^{\kappa-1}\left(b + \sum_{i=1}^{n} t_i^{\kappa}\right)^{-(a+p)}$$

En remplaçant a et b par les valeurs proposées dans l'énoncé on obtient une relation par rapport à une loi qu'on peut calculer mais on ne reconnaît pas de loi connue:

$$\pi\left(\kappa \mid t_{1:n}, p\right) \propto \pi(\kappa)\kappa^{p} \prod_{i=1}^{p} t_i^{\kappa-1} \left(10^{-3} + \sum_{i=1}^{n} t_i^{\kappa}\right)^{-(10^{-3}+p)}$$

Algorithme MCMC

On souhaite déterminer les lois de α et de κ. Pour cela on souhaite coder un algorithme MCMC qui va nous permettre d'avoir des achantillons distribués selon les deux densités recherchées. L'algorithme MCMC fonctionne en générant des chaînes de Markov qui explorent les espaces des valeurs possibles pour leur paramètre spécifique.

Il existe différentes méthodes pour déterminer les valeurs des élements de la chaîne. Ici, on choisit de méler deux méthodes:

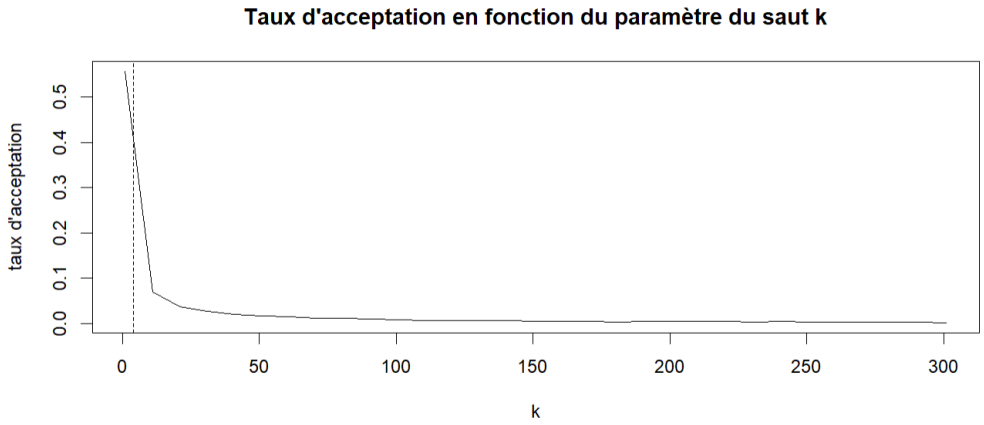
- Metropolis-Hastings pour κ. Metropolis-Hasting consiste à chaque étape à proposer une nouvelle valeur de la chaîne en utilisant une règle de transition basée sur la valeur actuelle de la chaîne. Cette proposition est acceptée ou rejetée en fonction de la probabilité d'acceptation, qui dépend de la valeur de la fonction de densité de probabilité de la distribution cible.
- Gibbs pour α. Gibbs consiste à chaque étape à proposer une nouvelle valeur de la chaîne en utilisant les distributions conditionnelles des paramètres tout en maintenant les autres paramètres fixés à leurs valeurs actuelles.

En premier lieu, on implemente une fonction qui va nous permettre de calculer la densité marginale à postériori de κ. On aura besoin de cette fonction pour calculer le ratio de MH. On la met sous forme log pour faciliter le calcul. Puis on implémente l'alogorithme de MH avec une boucle qui contient une première étape de mise à jour de κ et une deuxième étape de mise à jour de α.

Choix du saut k

La mise à jour de κ fait intervenir un paramètre de saut qu'il faut régler. Trop grand, les valeurs candidates serons toutes refusées, trop petit le déplacement de la chaîne dans l'espace serra trop lent. Pour choisir k, nous décidons de prendre le k qui permet d'avoir un taux d'acceptation de 40%. Nous traçons

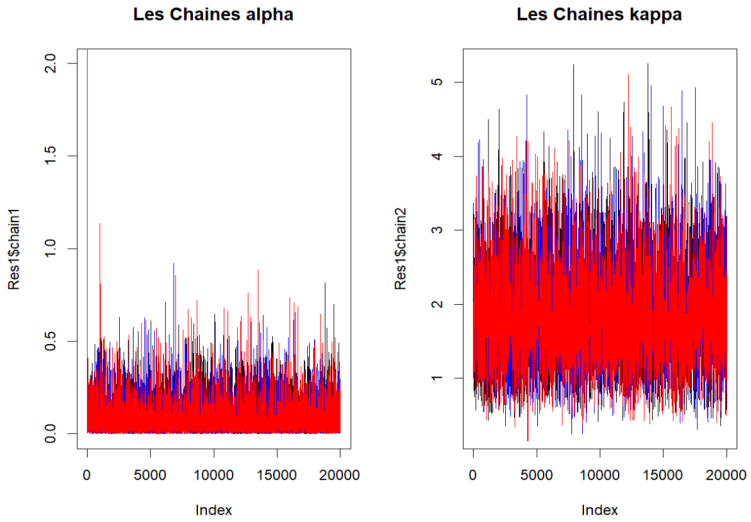
le taux d'acceptation pour différentes valeurs de saut et nous obtenons un saut optimale $k = 4.1$.



Execution des chaînes

Nous cherchons maintenant à s'assurer que nos chaînes convergent. Nous cherchons également à déterminer le nombre d'iteration à élmininer qui correspondent aux itération nécessaires avant stabilisation.

En premier lieux, nous observons visuellement les chaînes. Celles-ci se confondent après une période de chauffe donc il n'y a aucun indice de non convergence.

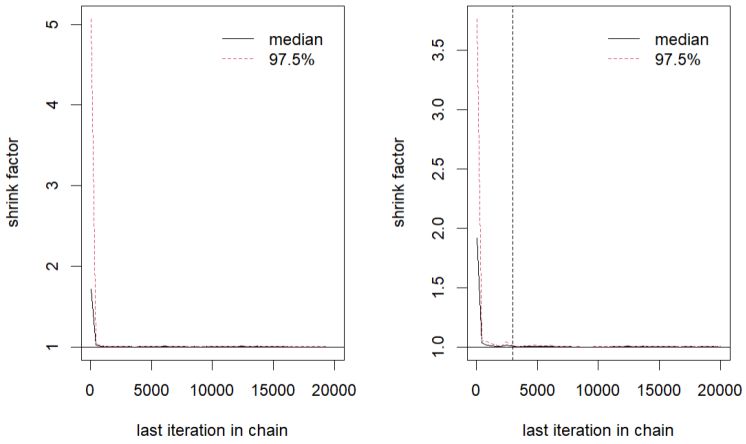


Représentation des chaînes de Markov supperposées

Ensuite, nous utilisons le critère de Gelman-Rubin pour avoir un indice suplementaire sur l'évenruelle convergence. Selon la règle standard : $R < 1.05+$ stabilité nous n'obtenons pas de problème de convergence majeur diagnostiqué selon ce critère.

```
# Potential scale reduction factors:
#
#   Point est. Upper C.I.
# [1,]          1          1
# [2,]          1          1
#
# Multivariate psrf
#
# 1
```

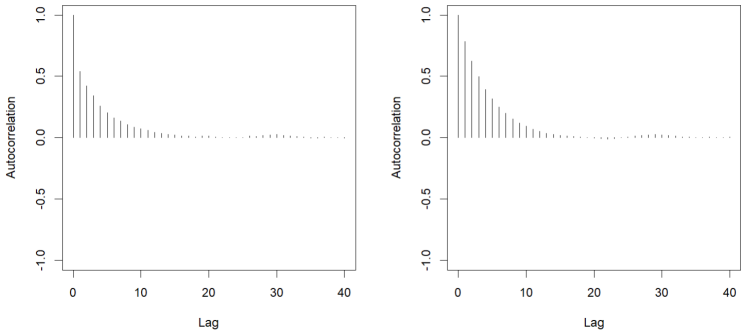
Ensuite nous plotons un graphique montrant l'évolution du facteur de réduction de Gelman et Rubin à mesure que le nombre d'itérations augmente. Nous en concluons que 20000 itérations dont 3000 itérations de temps de chauffe suffisent pour approcher correctement la loi a posteriori. Nous avons donc $G_0 = 3000$.



Graphique montrant l'évolution du facteur de réduction de Gelman et Rubin en fonction du nombre d'itérations

Analyse des autocorrélations intra-chaînes

Nous savons que plus les chaînes de Markov sont auto-corrélées plus il faudra générer de valeurs après le temps de chauffe pour bien approcher la loi a posteriori. On parle de "slow mixing". Ici, nous remarquons qu'il y a des corrélation donc il faudra faire plus de tirage dans la chaîne que de tirage indépendant que l'on souhaite.



Graphique des autocorrélations intra-chaînes

Détermination de la taille nécessaire de l'échantillon

Nous retirons le temps de chauffe qui n'est pas reprsentatif de tirages aléatoires dans la loi que nous cherchons à déterminer. Ensuite, afin de determiner la taille de la chaîne de Markov dont nous avons besoin, nous cherchons la taille d'échantillon effective (ESS) de l'échantillon a posteriori constitué. Nous obtenons des echantillons effectifs nettement plus faibles que les échantillons originaux ce qui est logique car on a observé des corrélations.

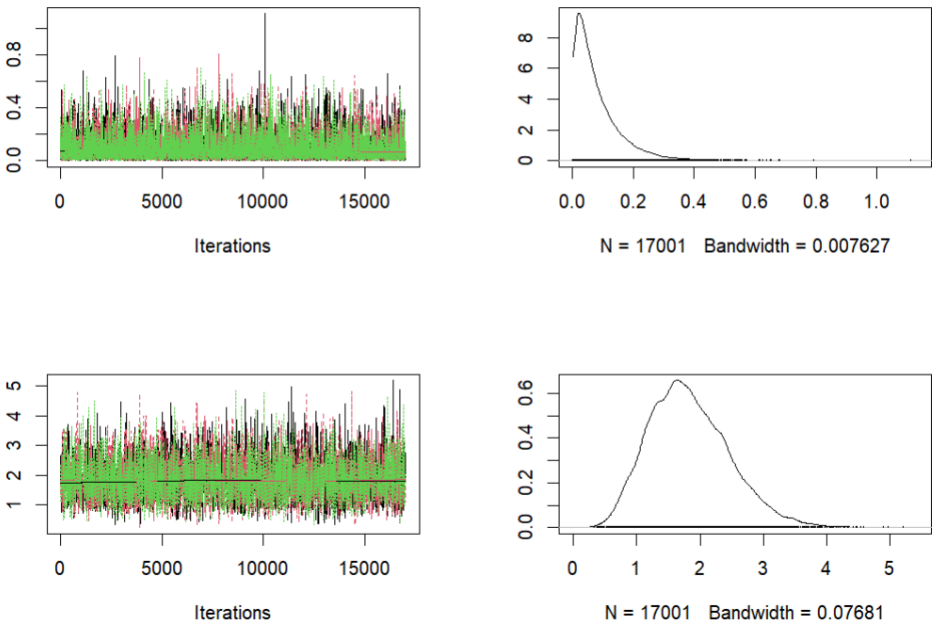
```
# var1      var2
# 7985.759  5705.583
```

Finalement, pour trancher sur la taille de l'échantillon, nous utilisons l'erreur de Monte-Carlo. Nous devons vérifier que Time-series SE (= Erreur de Monte-Carlo estimée) est inférieure à 5% de SD (= Ecart-type empirique a posteriori) pour chaque paramètre. Ici, l'écart est approprié on peut conserver cette taille d'échantillon.

Statistques et représentations des lois des paramètres

Finalement on obtient des échantillons simulant des tirages aléatoires dans les lois de α et κ. On affiche les statistiques correspondantes et on plot les densités correspondantes.

```
# Iterations = 1:17001
# Thinning interval = 1
# Number of chains = 3
# Sample size per chain = 17001
#
# Empirical mean and standard deviation for each variable,
# plus standard error of the mean:
#
#   Mean       SD  Naive SE Time-series SE
# [1,] 0.08376 0.07868 0.0003484      0.0008814
# [2,] 1.83277 0.63336 0.0028045      0.0083916
```



Plot des densités finales obtenues pour α et κ

Intervalles de crédibilité

La fonction summary nous permet d'obtenir directement les intervalles de crédibilité.

```
# 2. Quantiles for each variable:
#
#    2.5%      25%      50%      75%   97.5%
# [1,] 0.005394 0.02893 0.05986 0.1132 0.2956
# [2,] 0.775022 1.36898 1.77032 2.2313 3.2112
```

Conclusion

Dans le cadre de ce projet, nous avons exploré deux approches différentes pour estimer les paramètres d'une distribution et leurs densités postérieures. La première approche est fréquentiste, tandis que la seconde est basée sur l'inférence bayésienne. Dans l'approche bayésienne, nous avons utilisé l'algorithme MCMC (Chaînes de Markov par Monte Carlo) pour effectuer les estimations.
Après avoir appliqué les deux méthodes d'estimation (fréquentiste et bayésienne)

pour chaque facteur, nous avons constaté que les résultats obtenus étaient semblables. En effet, les premières valeurs estimées de α et κ étaient $\alpha = 0.08901503$ et $\kappa = 1.820281$ se situent aux point culminants des distributions de densité obtenues avec l'approche Bayésienne.
On remarque également que les deux approches, fréquentiste et bayésienne, diffèrent en termes de complexité des algorithmes utilisés. En effet, l'approche fréquentiste est plus simple et rapide, tandis que l'approche bayésienne, en raison de l'utilisation de l'algorithme MCMC, est plus complexe et plus coûteuse en termes de temps de calcul.