


# MCMC en pratique

# Logiciels MCMC

- **BUGS** : *Bayesian inference Using Gibbs Sampling*

1989 MRC BSU Université de Cambridge (UK)

⇒ logiciel flexible pour l'analyse bayésienne de modèles statistiques complexes à l'aide d'algorithmes MCMC



- WinBUGS : ⚠ clic-bouton + *Windows only* + développement arrêté  
<https://www.mrc-bsu.cam.ac.uk/software/bugs/the-bugs-project-winbugs/>
- OpenBUGS : ⚠ clic-bouton + *Windows only* + *Linux partiel*  
<https://www.mrc-bsu.cam.ac.uk/software/bugs/openbugs/>
- JAGS : 😊 ligne de commande interfacé avec   
<http://mcmc-jags.sourceforge.net/>

- **STAN**

<http://mc-stan.org/>

## rjags

Le logiciel JAGS est moderne et performant :

- s'appuie sur le langage BUGS pour spécifier un modèle bayésien
- interfacé avec  grâce au package rjags
- analyse des résultats dans  grâce aux packages
  - coda
  - HDInterval

# Convergence de Markov

Dans l'analyse bayésienne, on utilise les algorithmes MCMC pour générer un **échantillon de Monte-Carlo** de la loi *a posteriori*.

⇒ nécessite d'avoir atteint la **convergence** de la **chaîne de Markov** vers sa loi stationnaire (la loi *a posteriori*).

# Convergence de Markov

Dans l'analyse bayésienne, on utilise les algorithmes MCMC pour générer un **échantillon de Monte-Carlo** de la loi *a posteriori*.

⇒ nécessite d'avoir atteint la **convergence** de la **chaîne de Markov** vers sa loi stationnaire (la loi *a posteriori*).

⚠ *Aucune garantie sur la convergence en temps fini*

⇒ **étudier la convergence effective à chaque analyse**

# Convergence de Markov

Dans l'analyse bayésienne, on utilise les algorithmes MCMC pour générer un **échantillon de Monte-Carlo** de la loi *a posteriori*.

⇒ nécessite d'avoir atteint la **convergence** de la **chaîne de Markov** vers sa loi stationnaire (la loi *a posteriori*).

⚠ *Aucune garantie sur la convergence en temps fini*

⇒ **étudier la convergence effective à chaque analyse**

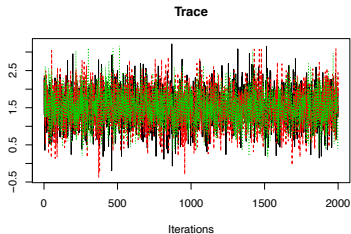
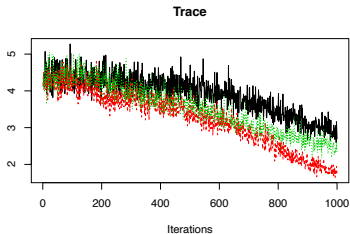
- 💡 Initialisation de **plusieurs chaînes de Markov** à partir de valeurs différentes
- ⇒ Si la **convergence est atteinte**, alors ces **chaînes doivent se confondre**

# Diagnostiques graphiques

- Trace
- Densité *a posteriori*
- Quantiles courants (*running quantiles*)
- Auto-corrélation
- Diagramme de Gelman-Rubin

# Trace

```
coda::traceplot()
```



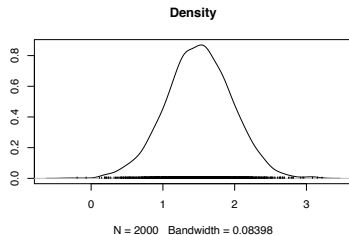
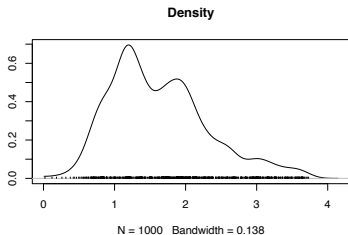
😊 Les chaînes doivent se superposer et se confondre

😞 ↗ n.iter et/ou ↗ phase de chauffe (*burn-in*)



# Densité

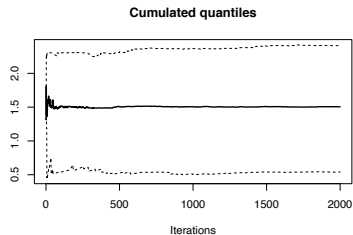
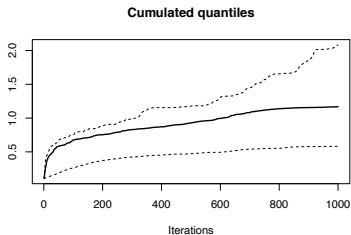
```
coda::densplot()
```



- 😊 Les densités doivent être bien lisses et uni-modales
- 😞 ↗ n.iter et/ou ↗ phase de chauffe (*burn-in*)

# Quantiles courants

```
coda::cumuplot()
```



- 😊 Les quantiles cumulés doivent être stables au cours des itérations
- 😞 ↗ n.iter et/ou ↗ phase de chauffe (*burn-in*)

# Statistique de Gelman-Rubin

- variation entre les différentes chaînes
- variation à l'intérieur d'une même chaîne

*Si l'algorithme a bien convergé, la variation inter-chaîne doit être proche de zéro*

$\theta_{[c]} = (\theta_{[c]}^{(1)}, \dots, \theta_{[c]}^{(N)})$  le  $N$ -échantillon de la chaîne  $c = 1, \dots, C$

La **statistique de Gelman-Rubin** :  $R = \frac{\frac{N-1}{N} W + \frac{1}{N} B}{W}$

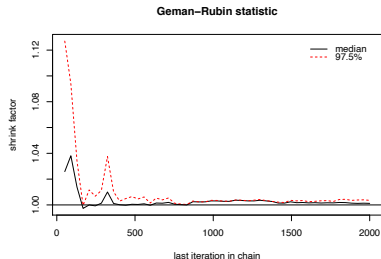
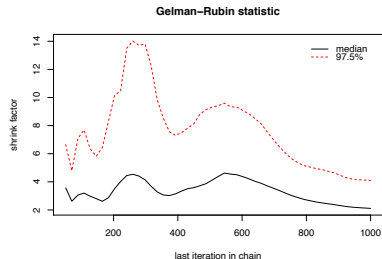
- variance inter-chaînes :  $B = \frac{N}{C-1} \sum_{c=1}^C (\bar{\theta}_{[c]} - \bar{\theta})^2$
- moyenne par chaîne :  $\bar{\theta}_{[c]} = \frac{1}{N} \sum_{t=1}^N \theta_{[c]}^{(t)}$
- moyenne globale :  $\bar{\theta} = \frac{1}{C} \sum_{c=1}^C \bar{\theta}_{[c]}$
- variance intra-chaîne :  $W = \sum_{c=1}^C s_{[c]}^2$ , avec  $s_{[c]}^2 = \frac{1}{N-1} \sum_{t=1}^N (\theta_{[c]}^{(t)} - \bar{\theta}_{[c]})^2$

$$N \rightarrow +\infty \ \& \ B \rightarrow 0 \Rightarrow R \rightarrow 1$$

D'autres statistiques existent. . .

# Diagramme de Gelman-Rubin

```
coda::gelman.plot()
```



😬 La médiane de la statistique de Gelman-Rubin doit se maintenir sous le seuil de 1,01 (voire 1,05)

😞 ↗ n.iter et/ou ↗ phase de chauffe (*burn-in*)

# Taille d'échantillon effective

Propriété de Markov  $\Rightarrow$  **auto-corrélation** entre les valeurs générées à la suite les unes des autres (échantillonnage dépendant) :

- diminue la quantité d'information disponible
- ralentit la convergence de la loi des grands nombres

**Taille d'échantillon effective** (*effective sample size*) quantifie cela :

$$ESS = \frac{N}{1 + 2 \sum_{k=1}^{+\infty} \rho(k)}$$

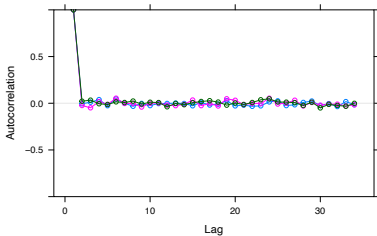
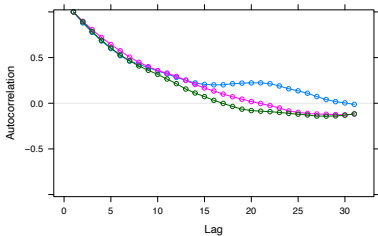
où  $\rho(k)$  désigne l'auto-corrélation avec *lag* de rang  $k$ .

*Espacer les échantillonnages conservés* (e.g. toutes les 2, 5, ou 10 itérations)

$\Rightarrow$  diminue la dépendance au sein de l'échantillon de Monte-Carlo généré

# Auto-corrélation

```
coda::acfplot()
```



😞 les auto-corrélations doivent décroître rapidement pour osciller autour de zéro

😞 ↗ thin et/ou ↗ n.iter et/ou ↗ phase de chauffe (*burn-in*)

# Erreur de Monte-Carlo

Pour un paramètre donnée, quantifie l'erreur introduite par la méthode de Monte-Carlo


- Cette erreur doit être la même d'une chaîne à l'autre
- Plus le nombre d'itérations  $N$  est grand, plus cette erreur de Monte-Carlo sera faible

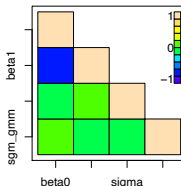
⚠ cette **erreur de Monte-Carlo** doit être faible au regard de la variance estimée de la loi *a posteriori*.

# Estimation

Grâce à un algorithme MCMC, on est capable d'obtenir un **échantillon de Monte-Carlo de la loi *a posteriori*** pour un **modèle bayésien**

On peut donc utiliser la **méthode de Monte-Carlo** pour obtenir des **estimations *a posteriori*** :

- Estimation ponctuelle (moyenne *a posteriori*, médiane *a posteriori*, ...)
- Intervalle de crédibilité (le plus étroit : *Highest Density Interval* – *HDI* via le package  `HDInterval`)
- Correlations croisées entre les paramètres
- ...





## Deviance Information Criterion (*DIC*)

La déviance s'écrit comme :  $D(\theta) = -2\log(p(\theta|\mathbf{y})) + C$  où  $C$  est une constante.

Le **Deviance Information Criterion** est alors :

$$DIC = \overline{D(\theta)} + p_D$$

où  $p_D = \left( D(\bar{\theta}) - \overline{D(\theta)} \right)$  représente une pénalité pour le nombre effectif de paramètres

⇒ Le *DIC* permet de comparer différents modèles sur les mêmes données  
*plus le DIC est bas, meilleur est le modèle !*

[M Plummer, *Penalized loss functions for Bayesian model comparison*, *Biostatistics*, 2008]