Markov chain convergence

In Bayesian analysis, MCMC algorithms are used to obtain a **Monte Carlo sample** from the *posterior* distribution

$\Rightarrow$ requires **Markov chain convergence** towards its stationary law (*posterior* distribution).

Markov chain convergence

In Bayesian analysis, MCMC algorithms are used to obtain a **Monte Carlo sample** from the *posterior* distribution

⇒ requires **Markov chain convergence** towards its stationary law (*posterior* distribution).

⚠ *No guaranty that this convergence will occur within finite time*

⇒ **study the convergence empirically for each analysis**

Markov chain convergence

In Bayesian analysis, MCMC algorithms are used to obtain a **Monte Carlo sample** from the *posterior* distribution

⇒ requires **Markov chain convergence** towards its stationary law (*posterior* distribution).

⚠ *No guaranty that this convergence will occur within finite time*

⇒ **study the convergence empirically for each analysis**

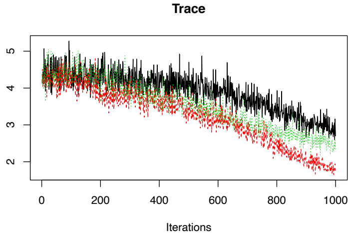
Initialisation of <u>**several**</u> **Markov chains** from different initial values
⇒ If **convergence is reached**, then these **chains must be overlapping**

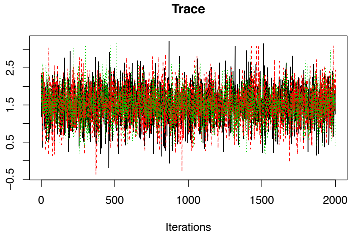
Graphical diagnoses

- Trace
- *Posterior* density
- Running Quantiles
- Auto-correlation
- Gelman-Rubin diagram

Trace

`coda::traceplot()`



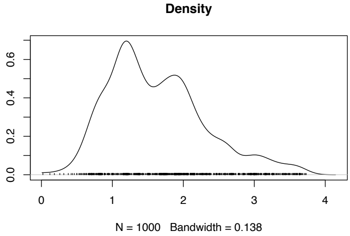


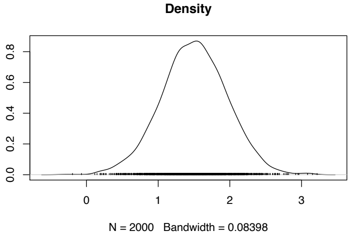
😁 chain traces must overlap and mix

☹️ $\nearrow$ `n.iter` and/or $\nearrow$ burn-in

Posterior density

`coda::densplot()`



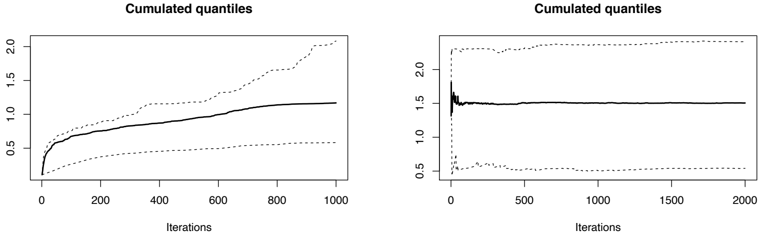


😁 density must be smooth and uni-modal

☹️ ↗ `n.iter` and/or ↗ burn-in

Quantiles courants

`coda::cumuplot()`



😁 running quantiles must be stable across iterations

☹️ ↗ `n.iter` and/or ↗ burn-in

Gelman-Rubin statistic

- variation between the different chains
- variation within a given chain

If the algorithm has properly converged, the between-chain variation must be close to zero

$\theta_{[c]} = (\theta_{[c]}^{(1)}, \dots, \theta_{[c]}^{(N)})$ the N-sample from chain number $c = 1, \dots, C$

Gelman-Rubin statistic: $R = \dfrac{\frac{N-1}{N} W \frac{1}{N} B}{W}$

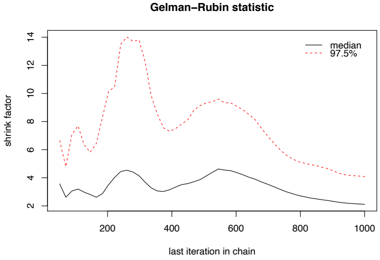
- between-chain variance: $B = \frac{N}{C-1} \sum_{c=1}^{C} (\bar{\theta}_{[C]} - \bar{\theta}.)^2$
- chain average: $\bar{\theta}_{[c]} = \frac{1}{N} \sum_{t=1}^{N} \theta_{[c]}^{(t)}$
- global average: $\bar{\theta}. = \frac{1}{C} \sum_{c=1}^{C} \bar{\theta}_{[C]}$
- within-chain variance: $s_{[c]}^2 = \frac{1}{N-1} \sum_{t=1}^{N} (\theta_{[c]}^{(t)} - \bar{\theta}_{[C]})^2$

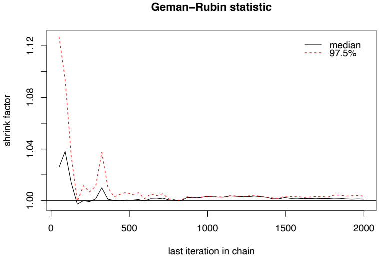
$$N \to +\infty \; \& \; B \to 0 \Rightarrow R \to 1$$

Other statistics exist...

Gelman-Rubin diagram

`coda::gelman.plot()`





😁 Gelman-Rubin statistic median must remain under the $1{,}01$ threshold (or $1{,}05$)

☹️ $\nearrow$ `n.iter` and/or $\nearrow$ burn-in

Effective Sample Size (ESS)

Markov property ⇒ **auto-correlation** between values sampled after one another (dependent sampling) :

- reduce the amount of information available within a sample size n
- slows down LLN convergence

Effective sample size quantifies this:

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

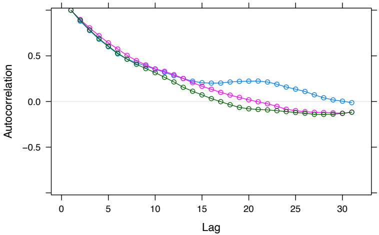
where $\rho(k)$ is the auto-correlation with lag k.

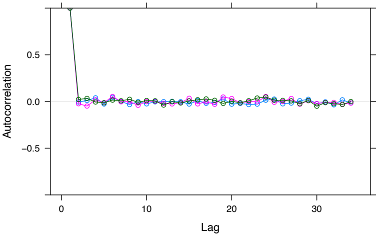
Space out saved samples (e.g. every 2, 5, or 10 iterations)

⇒ reduces dependency within the Monte Carlo sample generated

Auto-correlation

`coda::acfplot()`





- 😁 auto-correlations must decrease rapidly to oscillate around zero
- ☹️ ↗ `thin` and/or ↗ `n.iter` and/or ↗ burn-in

Monte Carlo error

For a given parameter, quantifies the error introduced through the Monte Carlo method

()standard deviation of the Monte Carlo estimator across the chains)

- That error must be consistent from one chain to another
- The larger N (number of iterations), the smaller the Monte Carlo error will be

⚠ This **Monte Carlo error** must be small with respect to the estimated variance of the *posterior* distribution