

# Introduction to the Bayesian framework in Biometrics

## Part II: Bayesian computation

Boris Hejblum

PhD Training course  
*Digital Public Health* program, University of Bordeaux

## Introduction

Estimating the *posterior* distribution  
is often costly

# Bayesian computational statistics

Computational aspects of Bayesian inference can get sophisticated but are key to its successful application

# Numerical integration – I

Real world applications:  $\boldsymbol{\theta} = (\theta_1, \dots, \theta_d)$

⇒ joint *posterior* distribution of all  $d$  parameters

⚠ hard to compute:

- complex likelihood
- integrating constant  $f(\mathbf{y}) = \int_{\boldsymbol{\theta}^d} f(\mathbf{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) d\boldsymbol{\theta}$
- ...

Analytical form rarely available

⇒ numerical computations: integral of  $d$  multiplicity  
– difficult when  $d$  is big (numerical issues as soon as  $d > 4$ )

# Numerical integration – II

Even dimension 1 can be tough !

## Example :

Let  $x_1, \dots, x_n$  *iid* according to a Cauchy distribution  $\mathcal{C}(\theta, 1)$  with *prior*  $\pi(\theta) = \mathcal{N}(\mu, \sigma^2)$  ( $\mu$  and  $\sigma$  known)

$$\begin{aligned} p(\theta|x_1, \dots, x_n) &\propto f(x_1, \dots, x_n|\theta)\pi(\theta) \\ &\propto e^{-\frac{(\theta-\mu)^2}{2\sigma^2}} \prod_{i=1}^n (1 + (x_i - \theta)^2)^{-1} \end{aligned}$$

⚠ normalizing constant has no analytical form  $\Rightarrow$  no analytical form for this *posterior* distribution

# Marginal *posterior* distributions

**Objective:** draw conclusion based on the joint *posterior* distribution

⇒ probability of all possible values for each parameter (i.e. their marginal distribution – uni-dimensional)

⚠ Recovering all of the *posterior* density **numerically** requires the calculation of multidimensional integrals **for each possible value of the parameter**

⇒ a sufficiently precise computation seems unrealistic

# Marginal *posterior* distributions

**Objective:** draw conclusion based on the joint *posterior* distribution

⇒ probability of all possible values for each parameter (i.e. their marginal distribution – uni-dimensional)

⚠ Recovering all of the *posterior* density **numerically** requires the calculation of multidimensional integrals **for each possible value of the parameter**

⇒ a sufficiently precise computation seems unrealistic

Algorithms based on **sampling simulations**  
especially **Markov chain Monte Carlo** (MCMC)

# Computational solutions

Bayes Theorem  $\Rightarrow$  *posterior* distribution



# Computational solutions

Bayes Theorem  $\Rightarrow$  *posterior* distribution

⚠ in practice:

- analytical form rarely available (very particular cases)
- integral to the denominator often very hard to compute

# Computational solutions

Bayes Theorem  $\Rightarrow$  *posterior* distribution

⚠ in practice:

- analytical form rarely available (very particular cases)
- integral to the denominator often very hard to compute

How can one estimate the *posteriori* distribution ?

$\Rightarrow$  sample according to this posterior distribution

- direct **sampling**
- **Markov chain Monte Carlo** (MCMC)

# Monte Carlo method

**Monte Carlo** : von Neumann & Ulam

(*Los Alamos Scientific Laboratory* – 1955)

⇒ use random numbers to compute quantities whose analytical computation is hard (or impossible)

# Monte Carlo method

## Monte Carlo : von Neumann & Ulam

(Los Alamos Scientific Laboratory – 1955)

⇒ use random numbers to compute quantities whose analytical computation is hard (or impossible)

- **Law of Large Numbers (LLN)**
- so-called “**Monte Carlo sample**”

⇒ compute various functions from that sample distribution

**Example :** One wants to compute  $\mathbb{E}[f(X)] = \int f(x)p_X(x)dx$

If  $x_i \stackrel{iid}{\sim} p_X$ ,  $\mathbb{E}[f(X)] = \frac{1}{N} \sum_{i=1}^N f(x_i)$  (LLN)

⇒ if one knows how to sample from  $p_X$ , one can then estimate  $\mathbb{E}[f(X)]$

...

# Monte Carlo method: illustration

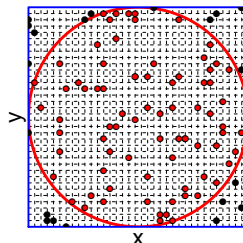
## $\pi$ estimation:

# Monte Carlo method: illustration

## $\pi$ estimation:



A casino roulette (in Monte Carlo ?)



A  $36 \times 36$  grid

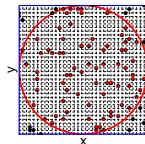
- 1 The probability of being inside the disk while in the square:  $p_C = \frac{\pi R^2}{(2R)^2} = \frac{\pi}{4}$
  - 2  $n$  points  $\{(x_{11}, x_{21}), \dots, (x_{1n}, x_{2n})\} = \{P_1, \dots, P_n\}$  on the  $36 \times 36$  grid (generated with the roulette)
  - 3 Count the number of points inside the disk
- ⇒ Compute the ratio (estimated probability of being inside the disk while in the square):  $\hat{p}_C = \frac{\sum P_i \in \text{circle}}{n}$

# Monte Carlo method: illustration

## $\pi$ estimation:



A casino *roulette* (in Monte Carlo ?)



A 36×36 grid

If  $n = 1000$  and 786 points are inside the disk :  $\hat{\pi} = 4 \times \frac{786}{1000} = 3.144$

One can improve the estimate by increasing:

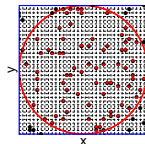
- the **grid resolution**, and also
- the number of points sampled  $n$ :  $\lim_{n \rightarrow +\infty} \hat{p}_C = p_C = \pi/4$  (LLN)

# Monte Carlo method: illustration

## $\pi$ estimation:



A casino roulette (in Monte Carlo ?)



A 36×36 grid

If  $n = 1000$  and 786 points are inside the disk :  $\hat{\pi} = 4 \times \frac{786}{1000} = 3.144$

One can improve the estimate by increasing:

- the **grid resolution**, and also
- the number of points sampled  $n$ :  $\lim_{n \rightarrow +\infty} \hat{p}_C = p_C = \pi/4$  (LLN)

**Monte Carlo** sample  $\Rightarrow$  compute various functions

e.g.  $\pi = 4 \times$  the probability of being inside the disk



# Your turn !



**Practical:** exercise 1

## Direct sampling methods

# Random & pseudo-random numbers

There exist several ways to generate so-called “random” numbers according to known distributions

**NB:** computer programs do not generate truly random numbers

Rather **pseudo-random**, which seem random but are actually generated by a deterministic process (depending on a “**seed**” parameter).

# Uniform sample generation

**Linear congruential algorithm:** sample pseudo-random numbers according to the Uniform distribution on  $[0, 1]$  (Lehmer, 1948)


- 1 Generate a sequence of integers  $y_n$  such as:

$$y_{n+1} = (ay_n + b) \bmod m$$

- 2  $x_n = \frac{y_n}{m-1}$

choose  $a$ ,  $b$  and  $m$  so that  $y_n$  has a long period &  $(x_1, \dots, x_n)$  can be considered *iid*

with  $y_0$  the seed

Remark:  $0 \leq y_n \leq m-1 \Rightarrow$  in practice  $m$  very large (e.g.  $2^{19937}$ , default in  which uses the Mersenne-Twister variation)

In the following, sampling pseudo-random numbers uniformly on  $[0, 1]$  will be considered reliable and used by the different sampling algorithms

## Other usual distributions

Relying on **relationships between the different usual distributions**  
starting from  $U_i \sim \mathcal{U}_{[0,1]}$

## Other usual distributions

Relying on **relationships between the different usual distributions**  
starting from  $U_i \sim \mathcal{U}_{[0,1]}$

Binomial  $Bin(n, p)$  :

$$Y_i = \mathbb{1}_{U_i \leq p} \sim \text{Bernoulli}(p)$$

$$X = \sum_{i=1}^n Y_i \sim Bin(n, p)$$

## Other usual distributions

Relying on **relationships between the different usual distributions** starting from  $U_i \sim \mathcal{U}_{[0,1]}$

Binomial  $\text{Bin}(n, p)$  :

$$Y_i = \mathbb{1}_{U_i \leq p} \sim \text{Bernoulli}(p)$$

$$X = \sum_{i=1}^n Y_i \sim \text{Bin}(n, p)$$

Normal  $\mathcal{N}(0, 1)$  (Box-Müller algorithm):

$U_1$  and  $U_2$  are 2 independent uniform variables on  $[0; 1]$

$$Y_1 = \sqrt{-2 \log U_1} \cos(2\pi U_2)$$

$$Y_2 = \sqrt{-2 \log U_1} \sin(2\pi U_2)$$

$\Rightarrow Y_1$  &  $Y_2$  are independent random variables each following a  $\mathcal{N}(0, 1)$

# Inverse transform sampling

**Definition:** For a function  $F$  defined on  $\mathbb{R}$ , its **generalized inverse** is defined as:  $F^{-1}(u) = \inf\{x \text{ tq } F(x) > u\}$



# Inverse transform sampling

**Definition:** For a function  $F$  defined on  $\mathbb{R}$ , its **generalized inverse** is defined as:  $F^{-1}(u) = \inf\{x \text{ tq } F(x) > u\}$

**Property:** Let

- $F$  be a cumulative probability distribution function
- $U$  be a uniform random variable on  $[0, 1]$

Then  $F^{-1}(U)$  defines a random variable with cumulative probability distribution function  $F$

If ① one knows  $F$ , the cumulative probability distribution function from which to sample

② one can invert  $F$

⇒ then one can sample this distribution from a uniform sample on  $[0, 1]$

# Inverse transform sampling: illustration

**Example:** sample from the Exponential distribution with parameter  $\lambda$

# Inverse transform sampling: illustration

**Example:** sample from the Exponential distribution with parameter  $\lambda$

- density of the Exponential distribution:  $f(x) = \lambda \exp(-\lambda x)$
- its cumulative probability distribution function (its integral):

$$F(x) = 1 - \exp(-\lambda x)$$

Let  $F(x) = u$

Then  $x = \dots$

# Inverse transform sampling: illustration

**Example:** sample from the Exponential distribution with parameter  $\lambda$

- density of the Exponential distribution:  $f(x) = \lambda \exp(-\lambda x)$
- its cumulative probability distribution function (its integral):  

$$F(x) = 1 - \exp(-\lambda x)$$

Let  $F(x) = u$

Then  $x = -\frac{1}{\lambda} \log(1 - u)$

$\Rightarrow$  and if  $U \sim U_{[0;1]}$ , then  $X = F^{-1}(U) = -\frac{1}{\lambda} \log(1 - U) \sim E(\lambda)$ .

# Your turn !



**Practical:** exercise 2

# Acceptance-rejection method

Use an **instrumental distribution**  $g$  (which we know how to sample from)  
 ⇒ to sample from the target distribution  $f$

The general principle is to **choose**  $g$  **close to**  $f$  and to propose samples from  $g$ , to accept some and reject others to get a sample following  $f$ .

# Acceptance-rejection method

Use an **instrumental distribution**  $g$  (which we know how to sample from)  
 $\Rightarrow$  to sample from the target distribution  $f$

The general principle is to **choose**  $g$  **close to**  $f$  and to propose samples from  $g$ , to accept some and reject others to get a sample following  $f$ .

Let  $f$  be the targeted density function

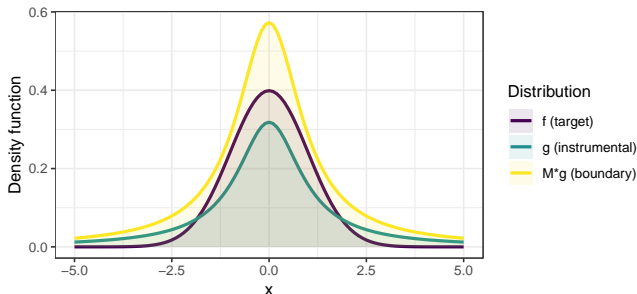
Let  $g$  be a proposal density function (from which one knows how to sample) such that, for all  $x$ :  $f(x) \leq Mg(x)$

While  $i \leq n$ :

- ① Sample  $x_i \sim g$  and  $u_i \sim \mathcal{U}_{[0,1]}$
- ② If  $u_i \leq \frac{f(x_i)}{Mg(x_i)}$ , **accept** the draw:  
 $y_i := x_i$   
 else **reject** it and return to 1.

$\Rightarrow (y_1, \dots, y_n) \stackrel{iid}{\sim} f$

# Acceptance-rejection: importance of the proposal



Example of a proposal and a target distribution for the accept-reject algorithm

**Remarque :** The smaller  $M$ , the greater acceptance rate

⇒ the more the algorithm is efficient at sampling from  $f$  (less iterations for a sample size  $n$ )

So one wishes  $g$  the as close as possible to  $f$  !

⚠  $g$  will necessarily have heavier tail than the target

⇒ when the number of parameters increases, acceptance rate decrease very rapidly (*curse of dimension*)



# MCMC Algorithms

# Markov chain definition

**Markov chain:** discrete time stochastic process

**Definition:** a series of random variables  $X_0, X_1, X_2, \dots$  (all valued over the same state space) with the “memoryless” **Markov property**:

$$p(X_i = x | X_0 = x_0, X_1 = x_1, \dots, X_{i-1} = x_{i-1}) = p(X_i = x | X_{i-1} = x_{i-1})$$

The set  $E$  of all possible values of  $X_i$  is called the **state space**

2 parameters:

- 1 initial distribution  $p(X_0)$
- 2 transition probabilities  $T(x, A) = p(X_i \in A | X_{i-1} = x)$

**NB:** only **homogeneous** Markov chains considered here:

$$p(X_{i+1} = x | X_i = y) = p(X_i = x | X_{i-1} = y)$$

# Markov chains properties

**Property**: a Markov chain is **irreducible** if all sets of non-zero probability can be reached from any starting point (i.e. any state is accessible from any other)

# Markov chains properties

**Property:** a Markov chain is **irreducible** if all sets of non-zero probability can be reached from any starting point (i.e. any state is accessible from any other)

**Property:** a Markov chain is **recurrent** if the trajectories  $(X_i)$  pass an infinite number of times in any set of non-zero probability of the state space

# Markov chains properties

**Property:** a Markov chain is **irreducible** if all sets of non-zero probability can be reached from any starting point (i.e. any state is accessible from any other)

**Property:** a Markov chain is **recurrent** if the trajectories  $(X_i)$  pass an infinite number of times in any set of non-zero probability of the state space

**Property:** a Markov chain is **aperiodic** if nothing induces periodic behavior of the trajectories

# Stationary law & ergodic theorem

**Definition:** A probability distribution  $\tilde{p}$  is called **invariant law** (or **stationary law**) for a Markov chain if it verifies the following property:  
if  $X_i \sim \tilde{p}$ , then  $X_{i+j} \sim \tilde{p} \forall j \geq 1$

*Remark:* a Markov chain can admit several stationary laws

# Stationary law & ergodic theorem

**Definition**: A probability distribution  $\tilde{p}$  is called **invariant law** (or **stationary law**) for a Markov chain if it verifies the following property:  
if  $X_i \sim \tilde{p}$ , then  $X_{i+j} \sim \tilde{p} \forall j \geq 1$

*Remark*: a Markov chain can admit several stationary laws

**Ergodic theorem** (infinite space): A positive irreducible and recurrent Markov chain admits a single invariant probability distribution  $\tilde{p}$  and converges towards it

# Markov chain example (discrete state space)– I

Doudou (a hamster) follows a Markov chain every minute with 3 states:

S sleep

E eat

W work out

⇒ its activity in 1min only depends on its current activity

Matrix of transition probabilities:

$$P = \begin{pmatrix} X_i/X_{i+1} & S & E & W \\ S & 0.9 & 0.05 & 0.05 \\ E & 0.7 & 0 & 0.3 \\ W & 0.8 & 0 & 0.2 \end{pmatrix}$$



# Markov chain example (discrete state space)– I

Doudou (a hamster) follows a Markov chain every minute with 3 states:

S sleep

E eat

W work out

⇒ its activity in 1min only depends on its current activity

Matrix of transition probabilities:

$$P = \begin{pmatrix} X_i/X_{i+1} & S & E & W \\ S & 0.9 & 0.05 & 0.05 \\ E & 0.7 & 0 & 0.3 \\ W & 0.8 & 0 & 0.2 \end{pmatrix}$$

- 1) Is the Markov chain irreducible ? recurrent ? aperiodic ?
- 2) Suppose Doudou is now asleep. What about in 2 min ? in 10 min ?
- 3) Suppose now that Doudou is working out. What about in 10 min ?

# Markov chain example (discrete state space) – II

1) Is the Markov chain irreducible ? recurrent ? aperiodic ?

...

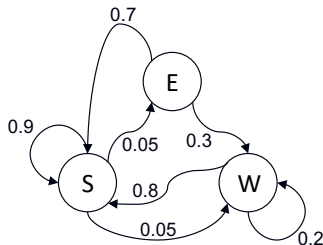
2) Suppose Doudou is now asleep. What about in 2 min ? in 10 min ?

...

3) Suppose now that Doudou is working out. What about in 10 min ?

# Markov chain example (discrete state space) – II

1) Is the Markov chain irreducible ? recurrent ? aperiodic ?



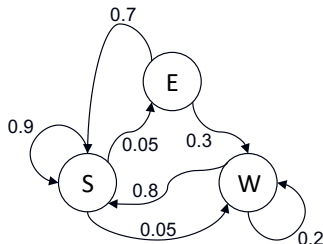
2) Suppose Doudou is now asleep. What about in 2 min ? in 10 min ?

...

3) Suppose now that Doudou is working out. What about in 10 min ?

# Markov chain example (discrete state space) – II

1) Is the Markov chain irreducible ? recurrent ? aperiodic ?



2) Suppose Doudou is now asleep. What about in 2 min ? in 10 min ?

$$x_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}^T \quad x_2 = x_0 P^2 = \begin{pmatrix} 0.885 \\ 0.045 \\ 0.070 \end{pmatrix}^T \quad x_{10} = x_2 P^8 = x_0 P^{10} = \begin{pmatrix} 0.884 \\ 0.044 \\ 0.072 \end{pmatrix}^T$$

## Markov chain example (discrete state space) – II

3) Suppose now that Doudou is working out. What about in 10 min ?

...

## Markov chain example (discrete state space) – II

3) Suppose now that Doudou is working out. What about in 10 min ?

$$x_0 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^T \quad x_{10} = x_0 P^{10} = \begin{pmatrix} 0.884 \\ 0.044 \\ 0.072 \end{pmatrix}^T$$

Here, the Markov chain being aperiodic, recurrent and irreducible, there is a stationary law:  $\tilde{p} = \tilde{p}P$ .

# MCMC algorithms: general principle

Approximate an integral (or another function) from a target distribution

# MCMC algorithms: general principle

Approximate an integral (or another function) from a target distribution

⇒ sample a Markov chain whose stationary law is the target (such as the *posterior*) distribution, then apply the Monte Carlo method.



# MCMC algorithms: general principle

Approximate an integral (or another function) from a target distribution

⇒ sample a Markov chain whose stationary law is the target (such as the *posterior*) distribution, then apply the Monte Carlo method.

Requires **two-fold convergence**:

- 1 the Markov chain must first converge to its stationary distribution:

$$\forall X_0, X_n \xrightarrow[n \rightarrow +\infty]{\mathcal{L}} \tilde{p}$$

# MCMC algorithms: general principle

Approximate an integral (or another function) from a target distribution

⇒ sample a Markov chain whose stationary law is the target (such as the *posterior*) distribution, then apply the Monte Carlo method.

Requires **two-fold convergence**:

- 1 the Markov chain must first converge to its stationary distribution:

$$\forall X_0, X_n \xrightarrow[n \rightarrow +\infty]{\mathcal{L}} \tilde{p}$$

- 2 then Monte Carlo convergence must also happen:

$$\frac{1}{N} \sum_{i=1}^N f(X_{n+i}) \xrightarrow[N \rightarrow +\infty]{} \mathbb{E}[f(X)]$$

# MCMC algorithms: general principle

Approximate an integral (or another function) from a target distribution

⇒ sample a Markov chain whose stationary law is the target (such as the *posterior*) distribution, then apply the Monte Carlo method.

Requires **two-fold convergence**:

- 1 the Markov chain must first converge to its stationary distribution:

$$\forall X_0, X_n \xrightarrow[n \rightarrow +\infty]{\mathcal{L}} \tilde{p}$$

- 2 then Monte Carlo convergence must also happen:

$$\frac{1}{N} \sum_{i=1}^N f(X_{n+i}) \xrightarrow{N \rightarrow +\infty} \mathbb{E}[f(X)]$$

$$\overbrace{X_0 \rightarrow X_1 \rightarrow X_2 \rightarrow \cdots \rightarrow X_n}^{\text{Markov chain convergence}} \rightarrow \overbrace{X_{n+1} \rightarrow X_{n+2} \rightarrow \cdots \rightarrow X_{n+N}}^{\text{Monte Carlo sample}}$$

# General framework of MCMC algorithms

MCMC algorithms uses an acceptance-rejection framework

- 1 Initialise  $x^{(0)}$
- 2 For  $t = 1 \dots n + N$  :
  - a Propose a new candidate  $y^{(t)} \sim q(y^{(t)} | x^{(t-1)})$
  - b Accept  $y^{(t)}$  with probability  $\alpha(x^{(t-1)}, y^{(t)})$ :  
 $x^{(t)} := y^{(t)}$   
 if  $t > n$ , "save"  $x^{(t)}$  (as part of the final Monte Carlo sample)

where  $q$  is the instrumental distribution for proposing new samples  
and  $\alpha$  is the acceptance probability.

# Choosing the instrumental distribution

**Not absolutely optimal choice** for the instrumental distribution  $q$   
proposing new samples

⇒ infinite possibilities: some better than others

# Choosing the instrumental distribution

**Not absolutely optimal choice** for the instrumental distribution  $q$  proposing new samples

⇒ infinite possibilities: some better than others

To guaranty convergence towards the target  $\tilde{p}$  :

- the support of  $q$  has to cover the support of  $\tilde{p}$
- $q$  must not generate periodic values

# Choosing the instrumental distribution

**Not absolutely optimal choice** for the instrumental distribution  $q$  proposing new samples

⇒ infinite possibilities: some better than others

To guaranty convergence towards the target  $\tilde{p}$  :

- the support of  $q$  has to cover the support of  $\tilde{p}$
- $q$  must not generate periodic values

**NB:** *ideally*  $q$  is **easy** and **fast** to **compute**

# Metropolis-Hastings algorithm

- 1 Initialise  $x^{(0)}$
- 2 For  $t = 1, \dots, n + N$  :
  - a Sample  $y^{(t)} \sim q(y^{(t)} | x^{(t-1)})$
  - b Compute the acceptance probability
$$\alpha^{(t)} = \min \left\{ 1, \frac{\tilde{p}(y^{(t)})}{q(y^{(t)} | x^{(t-1)})} \bigg/ \frac{\tilde{p}(x^{(t-1)})}{q(x^{(t-1)} | y^{(t)})} \right\}$$
  - c Acceptance-rejection step: sample  $u^{(t)} \sim \mathcal{U}_{[0;1]}$ 
$$x^{(t)} = \begin{cases} y^{(t)} & \text{if } u^{(t)} \leq \alpha^{(t)} \\ x^{(t-1)} & \text{else} \end{cases}$$

$$\alpha^{(t)} = \min \left\{ 1, \frac{\tilde{p}(y^{(t)})}{\tilde{p}(x^{(t-1)})} \frac{q(x^{(t-1)} | y^{(t)})}{q(y^{(t)} | x^{(t-1)})} \right\}$$

⇒ computable even if  $\tilde{p}$  is known only up to a constant !  
(like the posterior)



# Metropolis-Hastings: particular cases

Sometimes  $\alpha^{(t)}$  computation simplifies:

- **independent Metropolis-Hastings:**  $q(y^{(t)}|x^{(t-1)}) = q(y^{(t)})$
- **random walk Metropolis-Hastings:**  $q(y^{(t)}|x^{(t-1)}) = g(y^{(t)} - x^{(t-1)})$   
If  $g$  is symmetric ( $g(-x) = g(x)$ ), then:

$$\frac{\tilde{p}(y^{(t)})}{\tilde{p}(x^{(t-1)})} \frac{q(y^{(t)}|x^{(t-1)})}{q(x^{(t-1)}|y^{(t)})} = \frac{\tilde{p}(y^{(t)})}{\tilde{p}(x^{(t-1)})} \frac{\cancel{g(y^{(t)} - x^{(t-1)})}}{\cancel{g(x^{(t-1)} - y^{(t)})}} = \frac{\tilde{p}(y^{(t)})}{\tilde{p}(x^{(t-1)})}$$

# Pro and cons of Metropolis-Hastings

- 😊 very simple & very general
- 😊 allow sampling from uni- or multi-dimensional distributions
- 😞 choice of the proposal is crucial, but hard
  - ⇒ huge impact on algorithm performances
- 😞 quickly becomes inefficient dimension is too high

**NB:** a high rejection rate often implies important computation timings

# Simulated annealing

Change  $\alpha^{(t)}$  computation during the algorithm:

- 1  $\alpha^{(t)}$  must first be large to explore all of the state space
- 2 then  $\alpha^{(t)}$  must become smaller when the algorithm converges

# Simulated annealing

Change  $\alpha^{(t)}$  computation during the algorithm:

- ①  $\alpha^{(t)}$  must first be large to explore all of the state space
- ② then  $\alpha^{(t)}$  must become smaller when the algorithm converges

- ① Initialise  $x^{(0)}$
- ② For  $t = 1, \dots, n + N$  :
  - a Sample  $y^{(t)} \sim q(y^{(t)} | x^{(t-1)})$
  - b Compute the acceptance probability
$$\alpha^{(t)} = \min \left\{ 1, \left( \frac{\tilde{p}(y^{(t)})}{\tilde{p}(x^{(t-1)})} \frac{q(x^{(t-1)} | y^{(t)})}{q(y^{(t)} | x^{(t-1)})} \right)^{\frac{1}{T(t)}} \right\}$$
  - c Acceptance-rejection step: sample  $u^{(t)} \sim \mathcal{U}_{[0;1]}$ 
$$x^{(t)} := \begin{cases} y^{(t)} & \text{if } u^{(t)} \leq \alpha^{(t)} \\ x^{(t-1)} & \text{else} \end{cases}$$

# Simulated annealing

Change  $\alpha^{(t)}$  computation during the algorithm:

- 1  $\alpha^{(t)}$  must first be large to explore all of the state space
- 2 then  $\alpha^{(t)}$  must become smaller when the algorithm converges

- 1 Initialise  $x^{(0)}$
- 2 For  $t = 1, \dots, n + N$  :
  - a Sample  $y^{(t)} \sim q(y^{(t)} | x^{(t-1)})$
  - b Compute the acceptance probability
$$\alpha^{(t)} = \min \left\{ 1, \left( \frac{\tilde{p}(y^{(t)})}{\tilde{p}(x^{(t-1)})} \frac{q(x^{(t-1)} | y^{(t)})}{q(y^{(t)} | x^{(t-1)})} \right)^{\frac{1}{T(t)}} \right\}$$
  - c Acceptance-rejection step: sample  $u^{(t)} \sim \mathcal{U}_{[0;1]}$ 
$$x^{(t)} := \begin{cases} y^{(t)} & \text{if } u^{(t)} \leq \alpha^{(t)} \\ x^{(t-1)} & \text{else} \end{cases}$$

Ex:  $T(t) = T_0 \left( \frac{T_f}{T_0} \right)^{\frac{t}{n}} \Rightarrow$  particularly useful for avoiding local optima

# Gibbs sampler

When the dimension  $\nearrow \Rightarrow$  very hard to propose probable values

**Gibbs samplers:** re-actualisation coordinate by coordinate, while conditioning on the most recent values (no acceptance-rejection)

- ① Initialise  $x^{(0)} = (x_1^{(0)}, \dots, x_d^{(0)})$
- ② For  $t = 1, \dots, n + N$  :
  - a Sample  $x_1^{(t)} \sim p(x_1 | x_2^{(t-1)}, \dots, x_d^{(t-1)})$
  - b Sample  $x_2^{(t)} \sim p(x_2 | x_1^{(t)}, x_3^{(t-1)}, \dots, x_d^{(t-1)})$
  - c ...
  - d Sample  $x_i^{(t)} \sim p(x_i | x_1^{(t)}, \dots, x_{i-1}^{(t)}, x_{i+1}^{(t-1)}, \dots, x_d^{(t-1)})$
  - e ...
  - f Sample  $x_d^{(t)} \sim p(x_d | x_1^{(t)}, \dots, x_{d-1}^{(t)})$

**NB:** if the conditional distribution is unknown for some coordinates, an acceptance-rejection step can be included for this coordinate only (*Metropolis within gibbs*)

# Your turn !



**Practical:** exercise 3

## MCMC in practice



# MCMC softwares

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

1989 MRC BSU University of Cambridge (UK)

⇒ flexible software for Bayesian analysis in complex statistical models through MCMC algorithms



- WinBUGS: ⚠ clic + *Windows only* + stopped development  
<https://www.mrc-bsu.cam.ac.uk/software/bugs/the-bugs-project-winbugs/>
- OpenBUGS: ⚠ clic + *Windows only* + *Linux partially*  
<https://www.mrc-bsu.cam.ac.uk/software/bugs/openbugs/>
- JAGS: 😊 command line + R interface  
<http://mcmc-jags.sourceforge.net/>

- **STAN**: specialized for high-dimensional problems

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

## rjags

JAGS software is modern and efficient :

- relies on the BUGS language to specify a Bayesian model
-  interface thanks to rjags package
- results analysis with  packages
  - coda
  - HDInterval

# 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).

# 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 several Markov chains from different initial values

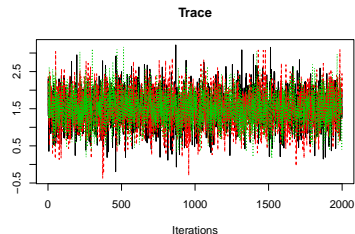
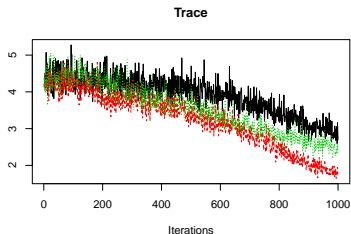
⇒ If **convergence is reached**, then these **chains must be overlapping**

# Graphical diagnostics

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

# Trace

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

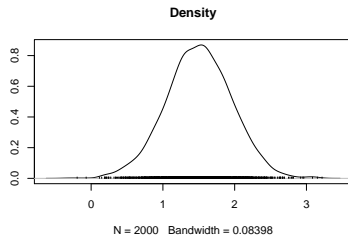
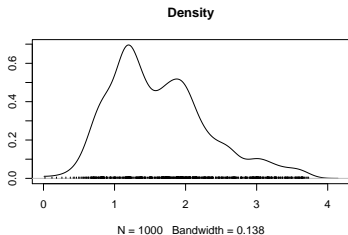


😊 chain traces must overlap and mix

😞 ↗ n.iter and/or ↗ burn-in

# Posterior density

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



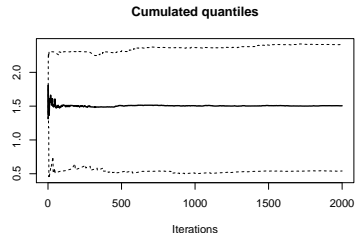
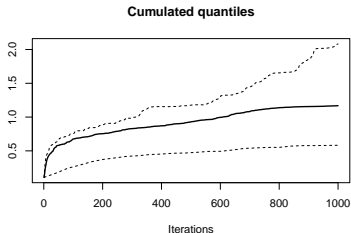
😊 density must be smooth and uni-modal

😞 ↗ n.iter and/or ↗ burn-in



# Running quantiles

```
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:**  $R = \frac{\frac{N-1}{N} W \frac{1}{N} B}{W}$

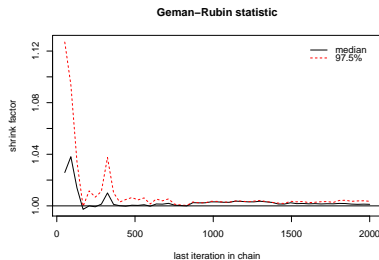
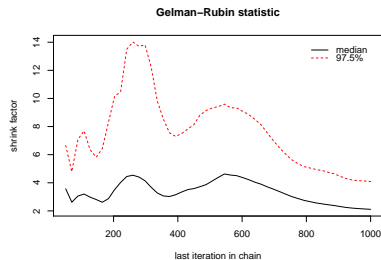
- between-chain variance:  $B = \frac{N}{C-1} \sum_{c=1}^C (\bar{\theta}_{[c]} - \bar{\theta})^2$
- chain average:  $\bar{\theta}_{[c]} = \frac{1}{N} \sum_{t=1}^N \theta_{[c]}^{(t)}$
- global average:  $\bar{\theta} = \frac{1}{C} \sum_{c=1}^C \bar{\theta}_{[c]}$
- within-chain variance:  $s_{[c]}^2 =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

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

Other statistics exist...

# Gelman-Rubin diagram

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



😊 Gelman-Rubin statistic median must remain under the 1,01 threshold (or 1,05)

😞 ↗ n.iter and/or ↗ burn-in

# Effective Sample Size (ESS)

Markov property  $\Rightarrow$  **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:

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

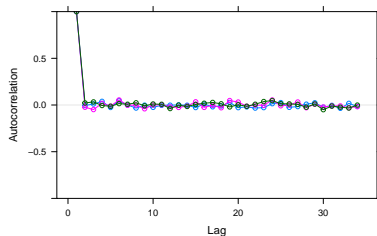
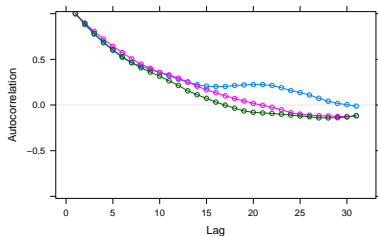
where  $\rho(k)$  is the auto-correlation with lag  $k$ .

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

$\Rightarrow$  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

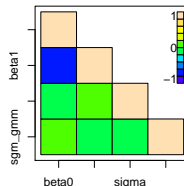
⚠ This **Monte Carlo error** must be small with respect to the estimated variance of the *posterior* distribution

# Estimation

Thanks to MCMC algorithms, one can obtain a **Monte Carlo sample from the *posterior* distribution** for a **Bayesian model**

**Monte Carlo method** can then be used to get ***posterior* estimates** :

- Point estimates (*posterior* mean, *posterior* median, ...)
- Credibility interval (shortest: *Highest Density Interval* – *HDI* with  package `HDInterval`)
- Correlations between parameters
- ...



## Deviance Information Criterion (*DIC*)

Deviance is:  $D(\theta) = -2\log(p(\theta|\mathbf{y})) + C$  with  $C$  a constant

**Deviance Information Criterion** is then:

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

where  $p_D = \left( \overline{D(\theta)} - \overline{D(\bar{\theta})} \right)$  represents a penalty for the effective number of parameters

⇒ *DIC* allows to compare different models estimated on the same data  
*the smaller the DIC, the better the model !*

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



# Your turn !



**Practical:** exercise 4

Questions ?

