Metropolis-Hastings and Gibbs Sampler algorithms

Alessandra Guglielmi

Politecnico di Milano
Dipartimento di Matematica
Milano, Italia
e-mail: alessandra.guglielmi@polimi.it

21 October 2020

Metropolis-Hastings algorithm - preliminaries

however suppose that we know $\pi(x)$ up to a constant (realistic since $\pi(\cdot)$ is the posterior)

METROPOLIS-HASTINGS chain components:

- **E** = state space (fixed) = parametric space in case $\pi(\cdot)$ is the posterior
- initial distribution of the chain (we're going to fix any initial distribution of the chain, we will start the chain from any point $x \in E$)

Goal: describe the transition kernel (how we go from X_n to X_{n+1}); the transition kernel is the rule according to which $X_n = x \rightsquigarrow X_{n+1} = y$

Suppose that the target distribution π has a density wrt a measure μ (e.g. Lebesgue); $\pi(x)$ = target density (from which we're not able to sample)

$E^+ := \{x \in E : \pi(x) > 0\}$ (support of $\pi(x)$); consider a transition probability $Q(x, dy) = q(x, y)\mu(dy)$ such that $Q(x; E^+) = 1 \ \forall x \notin E^+$, i.e. if we are not in E^+ at a certain time, but the chain evolves according to $Q(x, \cdot)$, we'll reach E^+ at the next iteration

Define

we can evaluate it because we know $\pi(\cdot)$ up to a constant (constant that will simplify)

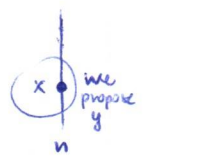
$$\alpha(x, y) := \begin{cases} \min\left(\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}, 1\right) & \text{if } \pi(x)q(x, y) > 0 \\ 1 & \text{if } \pi(x)q(x, y) = 0 \end{cases}$$

REM: $0 \le \alpha(x, y) \le 1$ and

$$\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)} \ge 1 \Leftrightarrow \pi(y)q(y, x) \ge \pi(x)q(x, y) \Leftrightarrow \frac{\pi(y)}{q(x,y)} \ge \frac{\pi(x)}{q(y,x)}$$

The Metropolis-Hastings algorithm

- $X_n = x$
- generate a candidate point $Y \sim Q(x, \cdot)$; then
  - accept the candidate point with probability $\alpha(x, y)$, so that $X_{n+1} = y$
  - reject y with probability $1 - \alpha(x, y)$ and set $X_{n+1} = x$



How do we simulate the rejection/acceptance? We simulate a $u([0,1]) \to u$. If $u \le \alpha(x,y)$ we accept, otherwise we reject.

How can we simulate this transition kernel? ←

If

$$p(x, y) := \begin{cases} q(x, y)\alpha(x, y) & \text{if } x \neq y \\ 0 & \text{if } x = y \end{cases}$$

then the transition probability that follows is

$$P(x, dy) = p(x, y)\mu(dy) + r(x)\delta_x(dy),$$

i.e.

$$P(x, A) = \int_A p(x, y)\mu(dy) + r(x)I_A(x) = \int_A q(x, y)\alpha(x, y)\mu(dy) + r(x)\mathbb{1}_A(x)$$

start from x and land in y, with $\forall y \to \mu(dy)$; remain in x with some prob. $r(x)$

Metropolis-Hastings algorithm

$$P(x, E) = \int_E p(x,y)\mu(dy) + r(x)\underbrace{\mathbb{1}_E(x)}_{=1} = 1$$ (since it's E and not A)

$\Longrightarrow$ $r(x)$ = probability of remaining at $x = \mathbb{P}(X_{n+1} = x | X_n = x)$

$$= 1 - \int_E q(x, y)\alpha(x, y)\mu(dy) \qquad = 1 - \int_E p(x,y)\mu(dy) > 0$$

We have to check that $(X)_n$ so defined is

1. irreducible (+ Harris-recurrent) → **notice that**: in order to prove this we have to specify our proposal density (and we're gonna consider 2 specific cases (next slides))
2. reversible wrt $\pi \rightsquigarrow$ let's check reversibility first; we already know that, in this case, π is the invariant distribution of this MC, but we are going to check it again... ←

REVERSABILITY (when we have densities):

$$p(x,y)\,\pi(x) = p(y,x)\,\pi(y)$$

(**proof.** at the end of the presentation (proof of the reversability)) *

Irreducibility of Metropolis-Hastings chains

In order to guarantee irreducibility, more assumptions on the proposal density $q(x, y)$ must be done

1. Random walk MH chain: $q(x, y) = f(y - x)$ where f is a density, i.e. $Y = x + Z$ with $Z \sim f$ $(Z \perp\!\!\!\perp x)$

   Th: If $f(x) > 0$ for all $x \in E = \mathbb{R}^p$, then $(X_n)_n$ is (π-)irreducible. Moreover, the MC is recurrent, and it is also aperiodic.

   REM: If the condition above is not valid, but $f(x) > 0$ for $x \in \mathcal{U}(0)$ and E^+ is an open connected set, then the conclusion of the Theorem holds.

   $f(x) > 0$ in a neighborhood of 0

   When f is symmetric, i.e. $f(x) = f(-x)$, then

$$\alpha(x, y) = \min\left(\frac{\pi(y)f(x-y)}{\pi(x)f(y-x)}, 1\right) = \min\left(\frac{\pi(y)}{\pi(x)}, 1\right)$$

they cancel if they're symmetric

Irreducibility of Metropolis-Hastings chains

2. Independence MH chain: $q(x, y) = f(y)$ where f is a density that does not depend on x , i.e. $Y \sim f$

   Th: If $f(x) > 0$ μ-a.e. on E^+, then $(X_n)_n$ is (π-)irreducible. Moreover, the MC is recurrent, and it is also aperiodic.

   irreducible if f is strictly positive on the support of $\pi(\cdot)$

   (just to know that exists, we'll focus more on the other case)

The Metropolis-Hastings algorithm

In general, for a MH chain $(X_n)_n$, we have that:

- if $r(x) > 0$ μ-a.e., then $(X_n)_n$ is aperiodic
- if $q(x, y) > 0$ for all $x, y \in E^+$, then $(X_n)_n$ is irreducible (since every subset of E^+ with positive Lebesgue measure can be reached in one step)
- If $(X_n)_n$ is irreducible, then $(X_n)_n$ is Harris-recurrent

Basically, in order to build a MH chain,

- we are required to sample from the density $q(x, \cdot)$
- we must be able to evaluate two ratios, $\frac{\pi(y)}{\pi(x)}$ and $\frac{q(y,x)}{q(x,y)}$

REM: the candidate point y is accepted if $\frac{\pi(y)}{q(x,y)} > \frac{\pi(x)}{q(y,x)}$

Gibbs Sampler - why?

When θ is multidimensional, using a MH algorithm would mean

- to sample from a Gaussian distribution that is multidimensional: NOT particularly efficient
- if the posterior is scaled quite differently or have pronounced skew or even multimodality in one dimension, use a joint proposal density: NOT particularly efficient

alternative algorithms with a *divide-and-conquer* nature: instead of sampling from a high-dim posterior, sample from low dimensional distributions

Gibbs Sampler - Bivariate target distribution

It is a MCMC algorithm where the target distribution is the posterior density, and we sample from univariate distributions

Target: generate $(X, Y) \sim \pi(x, y)$ when we know how to sample from

$$f_{X|Y} \text{ and } f_{Y|X}$$

Note: these two identify uniquely a joint distribution for X and Y

Algorithm:

- (X_0, Y_0) initial point
- If we know the MC (X_n, Y_n) at the n-step:

$$\text{sample } X_{n+1} \sim f_{X|Y}(\cdot|y_n)$$
$$\text{sample } Y_{n+1} \sim f_{Y|X}(\cdot|x_{n+1})$$

OUTPUT: $(X_n, Y_n)_{n\geq 0}$ is a bivariate MC

Gibbs Sampler - some theory

Theorem: π is the invariant distribution for $(X_n, Y_n)_{n\geq 0}$.

Definition: If $\pi_X(x) > 0$ and $\pi_Y(y) > 0$ implies $\pi(x, y) > 0$, then π is said to satisfy the positivity condition.

Equivalent to: $Supp(\pi_X) \times Supp(\pi_Y) \subset Supp(\pi)$.

Theorem: If π satisfies the positivity condition, then $(X_n, Y_n)_{n\geq 0}$ is (π-)irreducible. (and we need irreducibility for the Ergodic theorem to hold)

Theorem:
if $P((x, y); A \times B) = \int_{A\times B} f_{X|Y}(x_1|y) f_{Y|X}(y_1|x_1) dx_1 dy_1$, the transition probability of the chain, is absolutely continuous wrt the invariant distribution π of the MC, then $(X_n, Y_n)_{n\geq 0}$ is Harris-recurrent.

REM: the condition of the Th is stronger that the positivity condition; it will hold in all the "standard" examples

Gibbs Sampler

Summing up:

Theorem

If a Gibbs sampler MC $(X_n, Y_n)_n$ is π-irreducible and the transition probability $P(x; \cdot)$ is absolutely continuous wrt the invariant distribution π of the MC, then the MC is Harris-ergodic

and so we have all the nice properties

Basically, Gibbs sampler works when:

- $Supp(\pi_X) \times Supp(\pi_Y) = Supp(\pi)$.
- when $f_{X|Y}(\cdot|y)$, $f_{Y|X}(\cdot|x) > 0$ on the support sets of the marginals π_X, π_Y
- when π_X, π_Y exist (NO improper π).

priors and posteriors must be proper

Gibbs Sampler - Multivariate target distribution

Suppose the target density π is a p-dimensional density

Algorithm:

- $(X_1^{(0)}, \ldots, X_p^{(0)})$ initial point
- Suppose that, for $n = 1, 2, \ldots$, the MC at the n-step assumes value $x^{(n)} = (x_1^{(n)}, \ldots, x_p^{(n)})$; then

$$\text{sample } X_1^{(n+1)} \sim f_{X_1|X_2,\ldots,X_p}(\cdot|x_2^{(n)}, \ldots, x_p^{(n)})$$
$$\text{sample } X_2^{(n+1)} \sim f_{X_2|X_1,X_3\ldots,X_p}(\cdot|x_1^{(n+1)}, x_3^{(n)} \ldots, x_p^{(n)})$$
$$\vdots$$
$$\text{sample } X_p^{(n+1)} \sim f_{X_p|X_1,\ldots,X_{p-1}}(\cdot|x_1^{(n+1)}, \ldots, x_{p-1}^{(n+1)})$$

RESULTS: as for the bivariate case.

Gibbs Sampler as a particular case of MH

Gibbs sampler is a variant of MH, where each component of θ is updated sequentially and the transition probability is the product of the full-conditionals; the acceptance probability for each step (i.e. component) is always 1

If we cannot sample directly from one full-conditional (i.e. for those distributions known up to the normalizing constant): one step from a MH algorithm to sample from the full-conditional;

Metropolis-within-Gibbs

— Suppose that we're not able to sample from:

$f_{X_1|X_2,\dots,X_p}(\cdot|x_2,\dots,x_p)$

(maybe it's not one of the well known distributions and it does not exist in closed form)

⟹ We simulate one step from Metropolis-Hastings to sample the full conditional $(f_{X_1|X_2,\dots,X_p})$

Slide 5 : proof of reversability.

Thesis: $p(x,y)\,\pi(x) = p(y,x)\,\pi(y)$

proof.

Since α is a minimum we have 2 cases:

1. $\frac{\pi(y)\,q(y,x)}{\pi(x)\,q(x,y)} < 1$

$p(x,y)\,\pi(x) = q(x,y)\,\alpha(x,y)\,\pi(x) = \cancel{q(x,y)}\,\frac{\pi(y)\,q(y,x)}{\cancel{\pi(x)}\,\cancel{q(x,y)}}\,\cancel{\pi(x)} = q(y,x)\cdot\underbrace{1}_{:=\alpha(y,x)}\,\pi(y) \Longrightarrow$ and so we're done $(p(x,y)\pi(x) = p(y,x)\pi(y))$

Is it correct that $\alpha(y,x)=1$? Yes, :

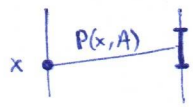
if $\frac{\pi(y)\,q(y,x)}{\pi(x)\,q(x,y)} < 1 \Longrightarrow \underbrace{\frac{\pi(x)\,q(x,y)}{\pi(y)\,q(y,x)}}_{L} > \underbrace{1}_{R} \Longrightarrow \alpha(y,x) = 1$ (since the minimum of L and R is achieved in 1)

2. $\frac{\pi(y)\,q(y,x)}{\pi(x)\,q(x,y)} > 1$

We start to work on $p(y,x)\,\pi(y)$ and with similar computations we prove the thesis. ■

The condition $p(x,y)\,\pi(x) = p(y,x)\,\pi(y)$ implies that $\pi(\cdot)$ is the invariant distribution of the chain.

Thesis: $\int_E P(x,A)\,\pi(x)\,\mu(dx) = \pi(A) \quad \forall A$



$\pi(A)$: we start from x and we land on A with transformation $P(x,A)$ and then we marginalize out w.r.t. the conditional distribution of $X_0 = x$.

$$\mathbb{P}(X_1 \in A) = \pi(A) = \int_E \mathbb{P}(X_1 \in A \,|\, X_0 = x)\,\underbrace{\chi_{X_0}(dx)}_{\text{"integrated w.r.t. the distribution of } X_0\text{"}}$$

proof.

$$\begin{aligned}
\int_E P(x,A)\,\pi(x)\,\mu(dx) &= \int_E \left(\int_A p(x,y)\,\mu(dy) + r(x)\,\mathbb{1}_A(x)\right)\pi(x)\,\mu(dx) \\
&= \int_E \left(\int_A p(x,y)\,\mu(dy)\right)\pi(x)\,\mu(dx) + \int_A r(x)\,\pi(x)\,\mu(dx) \\
&= \int_{y\in A}\left[\int_{x\in E} p(x,y)\,\pi(x)\,\mu(dx)\right]\mu(dy) + \int_A r(x)\,\pi(x)\,\mu(dx) \\
&\overset{\text{reversability}}{=} \int_{y\in A}\left[\int_{x\in E} p(y,x)\,\pi(y)\,\mu(dx)\right]\mu(dy) + \int_A r(x)\,\pi(x)\,\mu(dx) \\
&= \int_{y\in A}\pi(y)\,\underbrace{\left[\int_{x\in E} p(y,x)\,\mu(dx)\right]}_{1-r(y)}\mu(dy) + \int_A r(x)\,\pi(x)\,\mu(dx) \\
&= \int_{y\in A}\pi(y)\,\mu(dy) - \cancel{\int_{y\in A}\pi(y)\,r(y)\,\mu(dy)} + \cancel{\int_A r(x)\,\pi(x)\,\mu(dx)} \\
&= \pi(A)
\end{aligned}$$

■

Random Walk Metropolis-Hastings

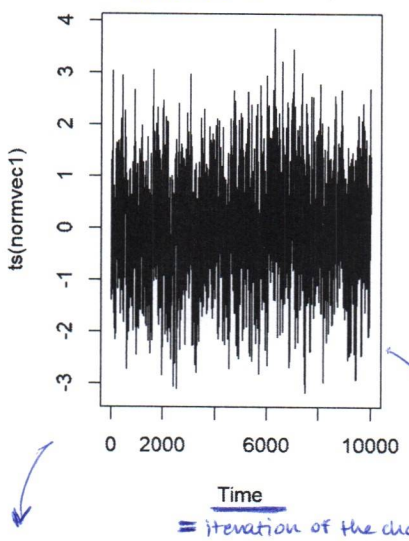
```
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
### EXAMPLE 1 - "simple" Metropolis-Hastings algorithm #
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
# Random Walk Metropolis-Hastings chain with invariant distribution = N(0,1);   ⟹ π ~ N(0,1)
# proposal density: q(x,y) = f(y-x), where f(u) = 1/(2alpha)I_(-alpha,alpha)),   Y = x + U ,  U ~ U([-α, α])
# i.e. Y = x+U(-alpha,alpha).
# The state space is E=\Re, an open connected set, and f(u)>0 in U(0).
# We'll run the algorithm under different values for alpha, e.g. alpha = 1,10,100
# The chain starts at 0.
# Tipically, for large values of alpha, the acceptance ratio will be small (i.e. we'll
# reject the candidate point often), since we're sampling the candidate point from
# a region where the exact posterior density is very low

norm<-function (n, alpha) {
        vec <- vector("numeric", n+1)
        x = 0      (initial point, actually x can be anything)
#       x <- -2*alpha  # RUN the algorithm by yourself with this value as the initial point!
        accept=0      ← # times that we accept the proposal density
        vec[1] <- x   ← vector which contains the output of the chain
        for (i in 2:n) {
                innov <- runif(1, -alpha, alpha)
                y <- x + innov        ← new proposed point
                aprob <- min(1, dnorm(y)/dnorm(x))      ← α(x,y)   (Notice: this time we know π(·))
                u <- runif(1)
# If condition "(u < aprob)" is NOT met, we'll skip command "x <- y",
# so that the MC does not move from x
                if (u < aprob) {
                        x <- y
                        accept1=accept+1
                        accept=accept1
                    }
                vec[i] <- x
        }
        vec[n+1]=(accept/n)   acceptance ratio
        vec
}

# Number of iterations of the MC
k = 10000

### ------------------------------------------------------------------------------
### alpha = 1
### ------------------------------------------------------------------------------
normvec  = norm(k,1)
normvec1 = normvec[1:k]     ← all the simulated values of the chain
rate1    = normvec[(k+1)]   ← acceptance rate
x11()
par(mfrow=c(1,2))
plot(ts(normvec1))
hist(normvec1,30, prob=T)
val=seq(-3,3,0.1)
points(val,dnorm(val),type='l',col='red')
```

do we accept? Only with probability $\alpha(x,y)$: we simulate a uniform $[0,1]$; if this is $< \alpha(x,y)$ then we accept, otherwise we reject

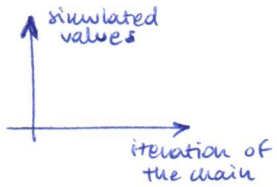
$f(\cdot)$ is not always >0 but it's >0 in a neighborhood of 0.

acceptance ratio → important number that we need to monitor to understand the best mixing of the chain

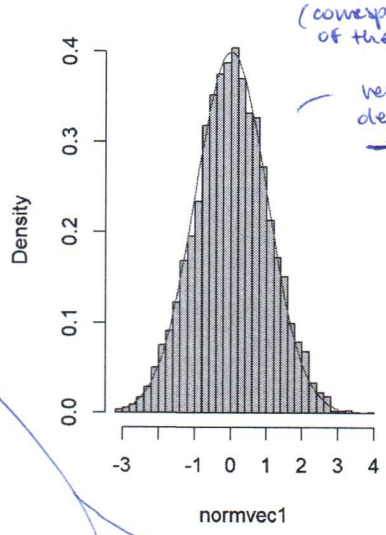
TRACEPLOT:



= iteration of the chain



Histogram of normvec1



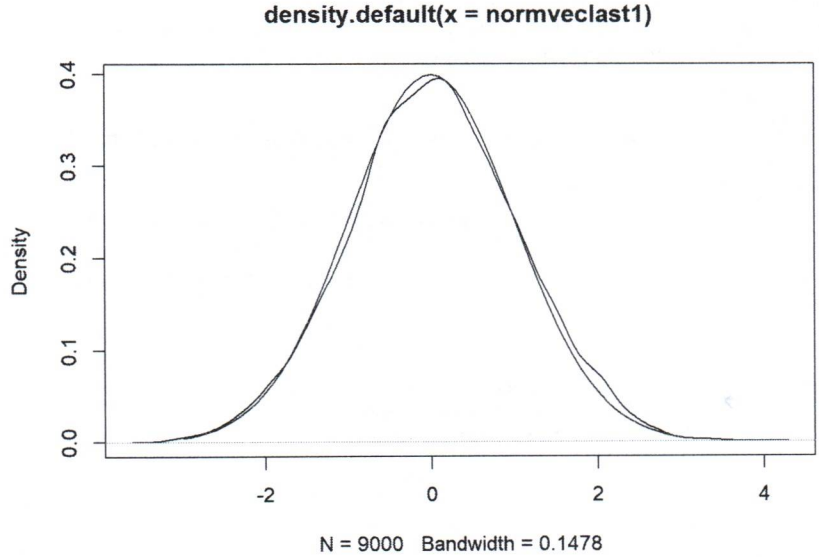
(corresponding histogram of the simulated values)

very close to the true density (standard normal) → good!

Here the chain explores the parameter space very efficiently, it covers the real line very efficiently (the support of the distribution should be $\mathbb{R}$, but we know that the std normal distr. concentrates its mass (more than 99%) in $[-3,+3]$, and we see that most of the draws are in that range)

It's clear that the simulated values are not independent, this is not Monte Carlo. There is correlation: we propose the new candidate as (old point) + something and if we reject we maintain the same value of the chain.

```
windows()
#plot(density(normvec1),bw='nrd')
#points(val,dnorm(val),type='l',col='red')

burnin=1000 # usually we get rid of the first burnin iterations,
            # where the chain hasn't reached stationarity yet
# The ergodic Th says we can use all iteration from any initial point x. However,
# we are averaging some function of some finite number of samples, so that out average will
# be a better approximation if we start at a typical point in the density we are sampling from
# and if this density is closer to the target distribution
b1=burnin+1
normveclast1=normvec[b1:k]

par(mfrow=c(1,1))
plot(density(normveclast1))
points(val,dnorm(val),type='l',col='red')
```

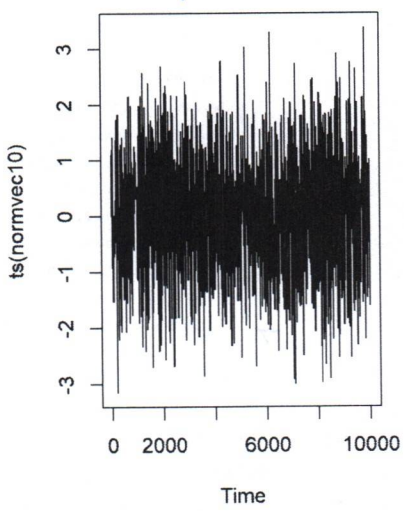


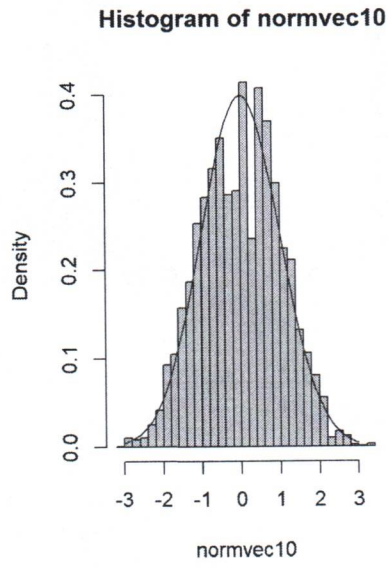

```
rate1
```

```
## [1] 0.8009
```
(kind of high, maybe too much)

```
### ------------------------------------------------------------------------------
### alpha = 10
### ------------------------------------------------------------------------------
normvec   = norm(k,10)
normvec10 = normvec[1:k]
rate10    = normvec[(k+1)]
x11()
par(mfrow=c(1,2))
plot(ts(normvec10))
hist(normvec10,30,prob=T)
points(val,dnorm(val),type='l',col='red')
```

Trace plot:





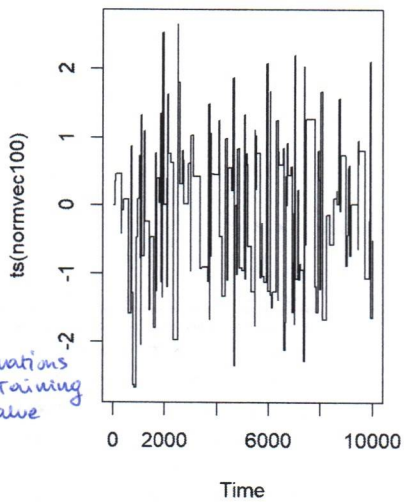
```
#windows()
#normveclast10=normvec10[b1:k] # traceplot discarding the first burnin iterations
#plot(density(normveclast10))
#points(val,dnorm(val),type='l',col='red')

rate10
```

```
## [1] 0.1603
```

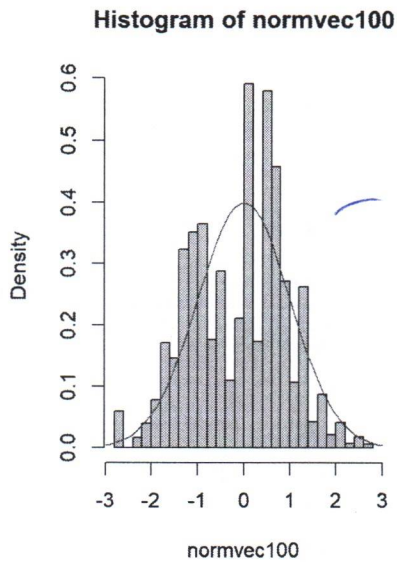
← much smaller than before: very often the chain keeps the same value (do not accept y)

```
### ----------------------------------------------------------------------------
### alpha = 100
### ----------------------------------------------------------------------------
windows()
normvec    = norm(k,100)
normvec100 = normvec[1:k]
rate100    = normvec[(k+1)]
par(mfrow=c(1,2))
plot(ts(normvec100))
hist(normvec100,30,prob=T)
points(val,dnorm(val),type='l',col='red')
```

Trace plot:



For many iterations we keep maintaining the same value



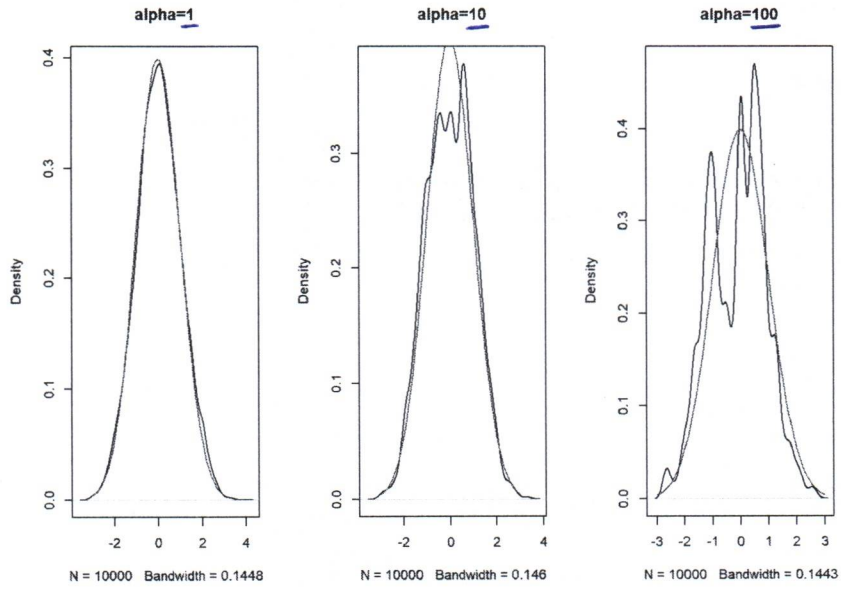
the histogram is really bad

```
#windows()
#normveclast100=normvec100[b1:k]  # traceplot discarding the first burnin iterations
#plot(density(normveclast100))
#points(val,dnorm(val),type='l',col='red')

rate100
```

```
## [1] 0.0164
```

much much smaller (< 2%): it's not so unusual to have something like that

```
### ----------------------------------------------------------------------------
### COMPARISON among kernel density plots of the simulated MCs for different values of alpha
### ----------------------------------------------------------------------------
windows()
par(mfrow=c(1,3))
plot(density(normvec1),main="alpha=1")
points(val,dnorm(val),type='l',col='red')
plot(density(normvec10),main="alpha=10")
points(val,dnorm(val),type='l',col='red')
plot(density(normvec100),main="alpha=100")
points(val,dnorm(val),type='l',col='red')
```



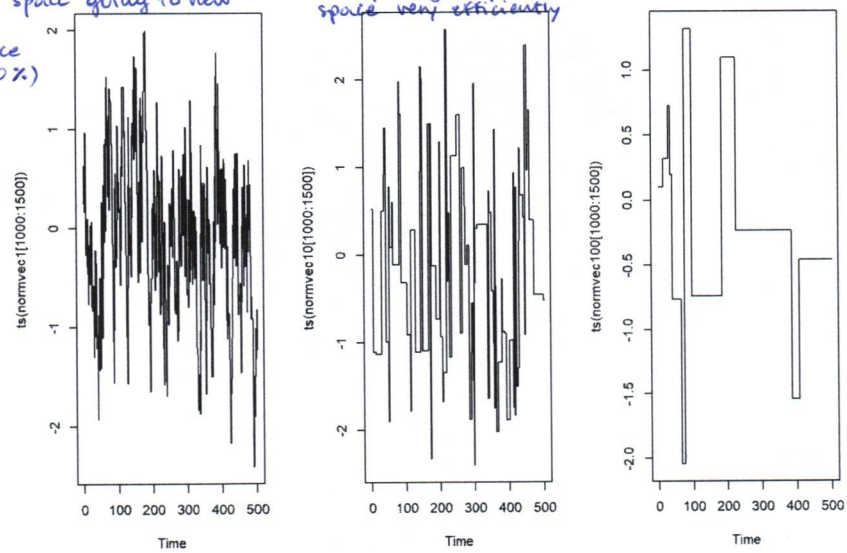

```
### ------------------------------------------------------------------------
### COMPARISON among traceplots of the simulated MCs for different values of alpha
### ------------------------------------------------------------------------
windows()
par(mfrow=c(1,3))
plot(ts(normvec1[1000:1500]))
plot(ts(normvec10[1000:1500]))
plot(ts(normvec100[1000:1500]))
```

we discard 1000 iterations

we accept the new candidate many times, we are exploring the state space going to new values (acceptance rate ~ 80%)

(accept. rate ~ 16%) we reject many times, we're not exploring the param. state space very efficiently

(acceptance rate < 2%)



However, why is it like this?
In case where α is very large we're proposing $x +$ something where the something can be very far from $[-3,3]$ (which is the subset of $\mathbb{R}$ in which the target distribution puts most of its mass). Using α very large we're proposing s.t. the ratio $\pi(y)/\pi(x)$ is very small (much smaller than 1) and therefore we will accept with very small probability.
If $\alpha = 1$ we do not move very much from $[-3,3]$ (that is "$\approx$" the support of the target distribution) and this is why the acceptance ratio is ~ 1.
However this is not good in general. If we propose y's too close to x we don't explore the support of $\pi(\cdot)$ very efficiently (since we remain often close to x).

```
### ------------------------------------------------------------------------
## COMPARISON among acceptance rates of the simulated MCs for different values of alpha
### ------------------------------------------------------------------------
rate1
```

A good acceptance rate should be $\in [0.23, 0.50]$
(It's a trade-off!)

```
## [1] 0.8009
```

```
rate10
```

```
## [1] 0.1603
```

```
rate100
```

```
## [1] 0.0164
```

Data are the heights of students in a campus.
The data are grouped in 6 classes.
We assume that the underneath model is gaussian with unknown mean and variance, but actually the data are grouped so ~~they're not iid~~ the likelihood is not a likelihood of iid gaussian distributed but it's the ~~multinomial~~ categorical.

Data description: $n = 211$ male students, $X_i = \text{height} \overset{iid}{\sim} N(\mu, \sigma^2)$ *

height	< 66 inches	[66,68)	[68,70)	[70,72)	[72,74)	≥ 74
freq.	14	30	49	70	33	15

$$\text{Likelihood} = p_1^{n_1} \cdot p_2^{n_2} \cdot \dots \cdot p_6^{n_6}$$

$$p_1 = \mathbb{P}(X_i < 66) = \phi\left(\frac{66-\mu}{\sigma}\right)$$

p_1 = mass given by * to the class "< 66"

$$p_2 = \mathbb{P}(X_i \in [66,68)) = \phi\left(\frac{68-\mu}{\sigma}\right) - \phi\left(\frac{66-\mu}{\sigma}\right)$$

$$p_6 = 1 - \sum_{i=1}^{5} p_i$$

$$\Longrightarrow \text{likelihood} = \left(\phi\left(\frac{66-\mu}{\sigma}\right)\right)^{n_1} \left(\phi\left(\frac{68-\mu}{\sigma}\right) - \phi\left(\frac{66-\mu}{\sigma}\right)\right)^{n_2} \dots$$

As a prior we consider:

$$\pi(\mu,\sigma) \propto \frac{1}{\sigma} \mathbb{1}_{(0,+\infty)}(\sigma) \qquad (\text{IMPROPER!!!})$$

and we consider a transformation of the parameter σ
(because if we keep it as it is then $\Theta = \mathbb{R} \times \mathbb{R}^+$ and in general use a MCMC method with a (somehow) bounded support, as here $\mathbb{R}^+$ instead of $\mathbb{R}$, is not good)

$$\Longrightarrow (\mu,\lambda) \in \Theta = \mathbb{R}^2, \quad \lambda := \log(\sigma)$$

$\Longrightarrow$ we re-parametrize the prior and we obtain a POSTERIOR as:

$$\pi(\mu,\lambda \mid \underbrace{n_1,\dots,n_6}) \propto L(\mu, e^{\lambda}, n_1,\dots,n_6) \cdot 1$$

the prior becomes proportional to a constant

In this way the posterior (a function only of μ and λ) is proportional to the likelihood, which we can evaluate.

$\Longrightarrow$ we know ratios of posteriors (we know the posteriors up to a normalizing constant)

$\Longrightarrow$ RWMH with $\underline{Y} = \underline{x} + \underline{Z}$; $\underline{Z} \sim N_2\left(\underline{0}, c^2 \left(\tilde{\mathbf{I}}\right)^{-1}\right)$

Random-Walk Metropolis-Hastings

what is $\tilde{I}$?
Generalized observed information matrix, we build it from:

$$- \frac{\partial^2}{\partial \mu \partial \lambda} \log\left(\pi(\mu, \lambda \mid n_1, \dots, n_6)\right)$$

```
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
### EXAMPLE 2 - Ex in 6.7 in Albert
### Section 6.7 Learning about a Normal Population from Grouped Data
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
# Example of a MH algorithm
rm(list=ls())
library(LearnBayes)
```

```
# Data are heights (in inches) of men from a local college
# The interest is in making inference about the mean and sd of the heights;
# X_i \sim n(\mu,\sigma^2)
# 1 inch= 2.54 cm
# However data are grouped: out of n=211 men, n_1=14 of them have height less than 66,
# n_2=30 have height between 66 and 68, ect.

# d is a list defining the intervals and the frequencies
d = list(int.lo=c(-Inf,seq(66,74,by=2)), int.hi=c(seq(66,74,by=2), Inf), f=c(14,30,49,70,33,15))
d
```

```
## $int.lo
## [1] -Inf   66   68   70   72   74
##
## $int.hi
## [1]  66  68  70  72  74 Inf
##
## $f
## [1] 14 30 49 70 33 15
```

```
help(groupeddatapost)
```

```
# it computes the log posterior density of (\mu,log \sigma) for normal sampling
# where the data is observed in grouped form.
# La prior per (mu,log(sigma)) è IMPROPRIA, proporzionale ad una cost.
# In pratica, è un modello multinomiale, in cui il parametro (= probabilità) vettoriale
# è la massa assegnata da una N(mu,sigma^2) ad una partizione di 6 intervalli

# INPUT: theta = valore di mu e log(sigma) in cui calcolo la log-posterior
#        data  = dataframe dei dati in forma raggruppata
# OUTPUT: log-posterior density

# EX: compute the log-posterior density at  mu=70, log(sigma)=1
groupeddatapost(c(70,1),d)
```

```
## [1] -348.416
```

```
help(laplace)
# It is an iterative method (Newton's method) to compute, for a general posterior density,
# an estimate of the posterior mode and the associated variance-covariance matrix,
# INPUT: the log-posterior density, the initial point of the iterative method,
#        parameters of the log-posterior density (the vector d in this case)

start=c(70,1) #punto iniziale - costruito da dati "fittizi"

fit=laplace(groupeddatapost,start,d)

fit
```

used to find $\tilde{I}$

(however no problem: we have convergence starting from whatever point in $\mathbb{R}^2$)

```
## $mode
## [1] 70.169880  0.973644
##
## $var
##               [,1]          [,2]
## [1,] 3.534713e-02 3.520776e-05
## [2,] 3.520776e-05 3.146470e-03
##
## $int
## [1] -350.6305
##
## $converge
## [1] TRUE
```

$\sim (\tilde{I}_n)^{-1}$

```
# fit$var is the matrix  (\tilde I_n)^{-1}, where \tilde I_n is the
# "generalized observed Fisher information matrix"
# (-d^2/ d d log(pi(mu,log(sigma)|x))  (d qui indica la DERIVATA!).
# Approximately the marginal posterior density of mu is N(70.17,0.035) and
# the marginal posterior density of lambda=log(sigma) is N(0.974, 0.003)

diag(fit$var)
```

```
## [1] 0.03534713 0.00314647
```

```r
modal.sds=sqrt(diag(fit$var))

# Otherwise use the function optim
?optim

### ------------------------------------------------------------------------------
### METROPOLIS-HASTINGS  ALGORITHM
### ------------------------------------------------------------------------------
# Build the MC according to a RANDOM WALK Metropolis algorithm:
# the bivariate proposal density q is N(val_prec,c^2 V)
# where the scale factor c=2, V  is the covariance matrix (\tilde I_n)^{-1};
# this is the covariance of the Gaussian approximation
# The PROPOSAL distribution has to be close to the true posterior, i.e.
#   it should be an approximation of the posterior!!!
# The scale factor c determines the behaviour of the MC
proposal = list(var=fit$var,scale=2)
proposal
```

← where we define our proposal $q(\cdot)$ (which in this case is: $\underline{Y} = \underline{x} + \underline{z}$, $\underline{z} \sim N_2(\underline{0}, c^2(\hat{I})^{-1})$)

Here we're proposing: $c = 2$ (after on we'll try to change it)

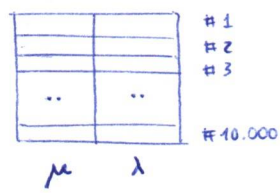
```
## $var
##              [,1]         [,2]
## [1,] 3.534713e-02 3.520776e-05
## [2,] 3.520776e-05 3.146470e-03
##
## $scale
## [1] 2
```

```r
# Proposal distributions: N(0,\Sigma), Uniform on the ball of radius R, t-Student
# (or more general mixtures of normals);
# usually, variance matrix of the Gaussian proposal is
# c^2 * inverse of the "generalized observed Fisher information" matrix
# for  c >0; here  c =1/2, 1, 2 work "well"

# La funzione del package LearnBayes è rwmetrop
# Simulates iterates of a random walk Metropolis chain for an
#   arbitrary real-valued posterior density defined by the user
# help(rwmetrop)

# print(rwmetrop)
# INPUT: log-posterior = groupeddatapost
#        proposal = N(0,c^2*V)
#        start= initial point of the MC
#        number of iterations=10000
#        d=parametri della log-posterior = iperparametri + dati
# the acceptance probability alpha(x,y) is the ratio of the posterior densities (random walk MH)

fit2 = rwmetrop(groupeddatapost,proposal,start,10000,d)

#OUTPUT :
# par    = a matrix of simulated values where each row corresponds to a simulated value of
#          the vector parameter
# accept = the acceptance rate of the algorithm

fit2
```

← $\in$ library(LearnBayes), which builds the Random Walk Metropolis-Hastings that we already saw in dimension 1

```
## $par
##              [,1]     [,2]
##     [1,] 70.10032 1.0972203
##     [2,] 70.10032 1.0972203
##     [3,] 70.10032 1.0972203
##     [4,] 70.04640 1.0100852
##     [5,] 70.04640 1.0100852
##     [6,] 70.04640 1.0100852
##     [7,] 70.04640 1.0100852
##     [8,] 70.04640 1.0100852
##     [9,] 70.04640 1.0100852
##    [10,] 70.04640 1.0100852
##    [11,] 70.04640 1.0100852
##    [12,] 70.03983 0.7984575
##    [13,] ..
```

← simulated values according to the chain for μ and λ ($\lambda = \log(\sigma)$)




```r
fit2$accept
```

```
## [1] 0.2959
```

(according to the range [0.23, 0.50] this is okay)

```
# The acceptance rate is good?
# The proposal should be more "diffuse" than posterior density
# in order to explore the whole support of the posterior density.
# If c is too small, the acceptance rate will be high; basically,
# a small scale c means that I'm sampling the candidate point y
# from a density with small variance, so that y will be close to x,
# and therefore alpha(x,y)=pi(y)/pi(x) will assume values close to 1.
# This means that the MC explores the state space weakly, i.e. the
# MC moves slowly through the state space, and the algorithm can
# turn out to be inefficient.
# On the other hand, if c is too large, alpha(x,y) will often be
# small, and many candidate points will be rejected,
# so that the overall acceptance rate r will be too small, and
# also in this case the algorithm is inefficient.
# Il tasso di accettazione BUONO per un RW-MH deve essere compreso
# in [0.25, 0.45] circa, o [0.23,0.5).
# Il valore 0.23... è il limite quando la dimensione dello spazio converge a infinito.

# For a random walk MH, remember that \alpha(x,y)=\pi(y)/\pi(x)
# indicates how probable the new proposed sample y is
# with respect to the current sample x.  If we attempt to move to
# a point that is more probable than the existing point (i.e.
# a point in a higher-density region of \pi), we will always
# accept the move. However, if we attempt to move to a less
# probable point, we will sometimes reject the move, and the
# more the relative drop in probability, the more likely
# we are to reject the new point.

# Per quanto riguarda independence chain Metropolis, tassi di
# accettazione che portano ad algoritmi efficienti sono
# decisamente più alti; teoricamente, più è alto il rapporto
# di accettazione e migliore è il tasso di convergenza della catena.

# Calcolo la media ergodica componente per componente
# help(apply)
# Medie a posteriori (calcolate col metodo MCMC) di mu e log(sigma)
post.means = apply(fit2$par,2,mean)
# Standard deviation delle 2 distribuzioni a posteriori marginali (MCMC)
post.sds   = apply(fit2$par,2,sd)
post.means
```

```
## [1] 70.1598596  0.9798509
```

```
post.sds
```

```
## [1] 0.18652908 0.05612245
```

in the theory we saw that we can approximate asymptotically the posterior with normal distr.

```
# Confronto fra medie e varianze dell'approx gaussiana con quelle a posteriori
cbind(c(fit$mode),modal.sds)
```

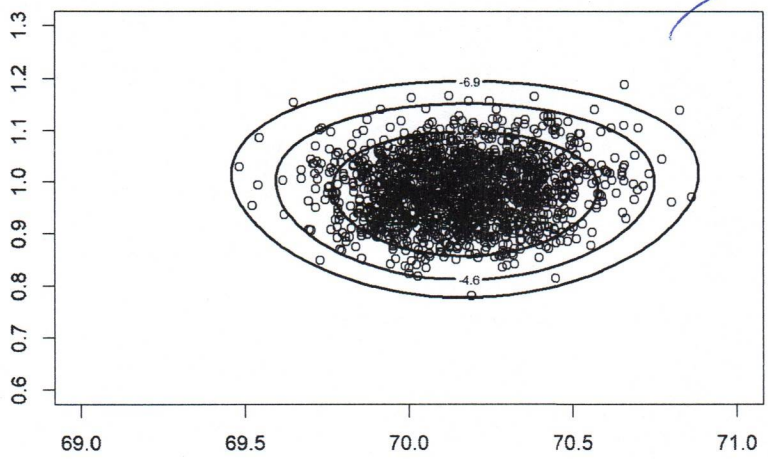
```
##                 modal.sds
## [1,] 70.169880 0.18800834
## [2,]  0.973644 0.05609341
```

```
cbind(post.means,post.sds)
```

```
##      post.means   post.sds
## [1,] 70.1598596 0.18652908
## [2,]  0.9798509 0.05612245
```

```
# In questo caso l'approssimazione col metodo di Laplace è buona

# Draw a countour plot of the true posterior (nota a meno della costante di normalizzazione)
#  with the MC points (from 5001-th on)
x11()
mycontour(groupeddatapost,c(69,71,.6,1.3),d)
points(fit2$par[5001:10000,1],fit2$par[5001:10000,2])
```

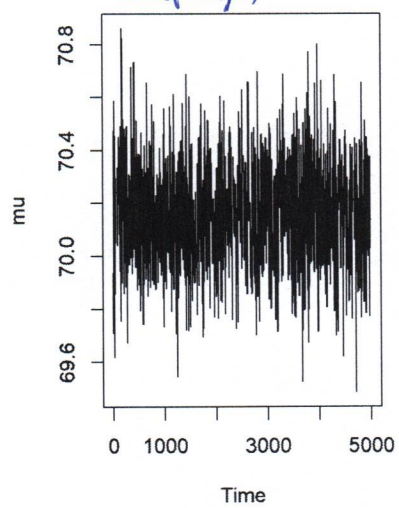
Notice that we (often) discard the first 5000 points because we're not sure that we already reached the convergence.



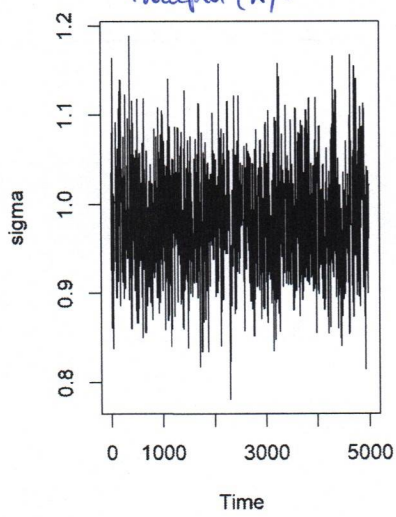
good: most of the points (simulated) are within the smallest ellipses (so we've hitting the proper target)

```
windows()
## marginal traceplots
par(mfrow=c(1,2))
plot(ts(fit2$par[5001:10000,1]),ylab="mu")
plot(ts(fit2$par[5001:10000,2]),ylab="sigma")
```

Traceplot (μ):



Traceplot (λ):



They seem so good!
(This* is the first visual check to see if 1. the chain has reached convergence and 2. the mixing of the chain is good enough)

* = traceplots + (contour plots & MCMC points)

```
# Change the proposal scale, e.g scale=0.1 and scale=10
```

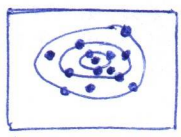
!

Scale = 10 ⇒ we've sampling the innovations from a normal which has a very large variance matrix, so potentially we're sampling candidates points very far from x. We expect the ratio $\pi(y)/\pi(x)$ to be very small and therefore the acceptance rate will be lower than before

⇒ [results] acceptance rate = 0.0214

Contourplot:

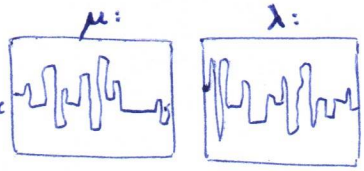
(many less points)



(the mixing is not very good)

Marginal traceplots:

(not thick, they're pieces of art, not good)



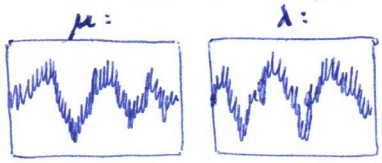
Scale = 0.1 ⇒ we've gonna accept very often (the innovation points are close to x, $\pi(y)/\pi(x) \sim 1$)

⇒ [results] acceptance rate = 0.9448

Contourplot:

too much! moreover we're not exploring efficiently the parameter state space (very small moves)

Marginal traceplots:



not so bad as traceplots, however the correlation is too high

Suppose we want to estimate: $\int h(\theta)\ \pi(d\theta|\underline{x}) = \mathbb{E}_\pi[h(\theta)|\underline{x}]$.

We simulate $\theta^{(1)}, \dots, \theta^{(T)}$ from the MCMC (in particular Metropolis-Hastings) and consider the estimator for $\mathbb{E}_\pi[h(\theta)|\underline{x}]$ that is:

(the Ergodic mean): $\bar{h}^{(T)} = \frac{1}{T}\sum_{t=1}^{T} h(\theta^{(t)})$.

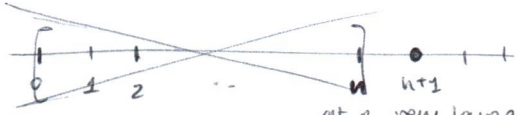
We know that the variance of this estimator is: $\mathrm{Var}\left(\bar{h}^{(T)}\right) = \frac{1}{T}\sigma_h^2$ $\quad\left(\sigma_h^2 = \mathrm{Var}_\pi(f(X_0)) + 2\sum_{k=1}^{\infty}\mathrm{Cov}_\pi(f(X_0), f(X_k))\right.$

We call the MCMC standard error $:= \sqrt{\mathrm{Var}(\bar{h}^{(T)})} = \sqrt{\frac{1}{T}\sigma_h^2}$ $\quad\left.X_0, X_k \text{ marginal} \sim \pi\right)$.

and we need to control it because it says what is the error probability when we estimate $\mathbb{E}_\pi[h(\theta)|\underline{x}]$ with $\bar{h}^{(T)}$.

How can we estimate σ_h^2 with MCMC?

Suppose we translate $\theta^{(1)}, \dots, \theta^{(T)}$ so much that they're already distributed according to π.



at a very large n we'll have that the conditional distribution of the chain (given the initial point) will be very close to the limit distribution (which is our posterior)

$\Rightarrow$ consider only the draws simply ~~about~~ from $n+1$ ~~and re-numbering~~

In this way, also re-nomining $n+1 := 1$, we'll have that the chain already has as marginal distribution the posterior (the limit distribution)

* MONITORING THE CONVERGENCE TO THE STATIONARY DISTRIBUTION

$$\sigma_h^2 = T \cdot \mathrm{Var}(\bar{h}^{(T)}) = T\,\mathrm{Var}\left(\frac{1}{T}\sum_{t=1}^{T} h(\theta^{(t)})\right)$$

$$= \frac{1}{T}\Big[\mathrm{Var}(h(\theta^{(1)})) + \mathrm{Var}(h(\theta^{(2)})) + \dots + \mathrm{Var}(h(\theta^{(T)})) +$$
$$+ 2\,\mathrm{Cov}(h(\theta^{(1)}), h(\theta^{(2)})) + 2\,\mathrm{Cov}(h(\theta^{(1)}), h(\theta^{(3)})) + \dots + 2\,\mathrm{Cov}(h(\theta^{(1)}), h(\theta^{(T)})) +$$
$$+ 2\,\mathrm{Cov}(h(\theta^{(2)}), h(\theta^{(3)})) + \dots + 2\,\mathrm{Cov}(h(\theta^{(2)}), h(\theta^{(T)})) +$$
$$+ \dots + \dots + \dots +$$
$$+ 2\,\mathrm{Cov}(h(\theta^{(T-1)}), h(\theta^{(T)}))\Big]$$

marginally those θ's are iid, since they're sampled from the posterior distribution. All these variances are equal and we call these variances σ^2

$$= \frac{1}{T}\left[T\sigma^2 + 2\sum_{j=1}^{T-1}(T-1)\,\mathrm{Cov}(h(\theta^{(1)}), h(\theta^{(j+1)}))\right]$$

$$= \sigma^2 + 2\sum_{j=1}^{T-1}\left(1-\frac{j}{T}\right)\underbrace{\mathrm{Cov}(h(\theta^{(1)}), h(\theta^{(j+1)}))}_{:=\gamma_j}$$

$$\rho_j := \frac{\gamma_j}{\sigma^2} \Longrightarrow \gamma_j = \sigma^2\rho_j \qquad \sigma^2 = \sqrt{\mathrm{Var}(h(\theta^{(\cdot)}))}\cdot\sqrt{\mathrm{Var}(h(\theta^{(*)}))}$$

$$= \sigma^2 + 2\sum_{j=1}^{T-1}\left(1-\frac{j}{T}\right)\sigma^2\rho_j$$

$$= \sigma^2\left(1 + 2\underbrace{\sum_{j=1}^{T-1}\left(1-\frac{j}{T}\right)\rho_j}_{\approx \sum_{j=1}^{\infty}\rho_j}\right) \Longrightarrow \boxed{\sigma_h^2 \approx \sigma^2\left(1 + 2\sum_{j=1}^{\infty}\rho_j\right)}$$

$(\sigma_h^2 > \sigma^2)$

↳ variance of the estimator *if* the draws were ⫫

So if we're able to approximate σ^2 and ρ_j then we're able to approximate σ_h^2 (through the MCMC) and we can control the MC standard error

↳ but they're not ⫫, it's MCMC, not MC

$$\hat{\sigma}^2 = \frac{1}{T}\sum_{j=1}^{T}\left(h(\theta^{(j)}) - \bar{h}^{(T)}\right)^2, \qquad \hat{\gamma}_j = \frac{1}{T}\sum_{j=1}^{T-j}\left(h(\theta^{(i)}) - \bar{h}^{(T)}\right)\left(h(\theta^{(i+j)}) - \bar{h}^{(T)}\right) \quad \forall i \text{ fixed (w.l.o.g. } i=1)$$

$$\hat{\rho}_j = \frac{\hat{\gamma}_j}{\hat{\gamma}_0} = \frac{\hat{\gamma}_j}{\hat{\sigma}^2} \Longrightarrow \boxed{\hat{\sigma}_h^2 = \hat{\sigma}^2\left(1 + \sum_j \hat{\rho}_j\right)}$$

What is the effective sample size? ($\hat{n}$)

$$\widehat{\mathrm{Var}}(\bar{h}^{(T)}) = \frac{\hat{\sigma}_h^2}{T} = \frac{\hat{\sigma}^2}{\hat{n}} \Longrightarrow \hat{n} = \frac{\hat{\sigma}^2}{\hat{\sigma}_h^2/T} = \frac{\hat{\sigma}^2}{\hat{\sigma}_h^2}T < T \Longrightarrow \boxed{\hat{n} < T}$$

The larger is $\hat{n}$ the less autocorrelated are the chains ($\hat{n} \sim T \Rightarrow$ the draws are almost iid)

```
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
### Section 6.8 Example of Output Analysis
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
# MONITORING the CONVERGENCE of the MC to the STATIONARY distribution
# If we simulate realizations from a MC, then under 'broad' conditions,
# EVENTUALLY the simulated values will be marginally distributed
# acconding to the invariant distr. and the ergodic means will
# converge to correspondent integrals of the invariant distribution.
# However, initial points of the MC, if included in the
# computation of the ergodic mean, may influence the value of the
# ergodic mean itself, yielding a poor approximation.
# For this reason we usually discard the first non-stationary
# portion of the chain, called BURN-IN.
# Therefore we need to be able to detect when the marginal behaviour
# of the MC is sufficiently close to stationarity, and harvest all
# subsequent realizations as a DEPENDENT sample from the stationary distribution.
# We diagnose convergence retrospectly, by guessing for how long
# to run the simulation and then trying to determinate if some
# latter portion of the chain can be considered stationary.
# The sample size required depends on the EFFICIENCY of the MC;
# moreover this sample size increases with an increasing level
# of serial dependence of the chain (autocorrelation of the MC).
rm(list = ls())
library(LearnBayes)
d = list(int.lo=c(-Inf,seq(66,74,by=2)), int.hi=c(seq(66,74,by=2), Inf), f=c(14,30,49,70,33,15))
library(coda)
```

If the autocorrelation is high we're gonna consider more samples

```
library(lattice)

start = c(70,1)
fit   = laplace(groupeddatapost,start,d)

# Let us choose an UNLUCKY initial point for the chain,
# and a small scale factor for the proposal c=0.2 ;
# MC will move slowly through the state space since candidate y
# will be close to current x
start=c(65,1)
proposal = list(var=fit$var,scale=0.2)
bayesfit = rwmetrop(groupeddatapost,proposal,start,10000,d)

bayesfit$accept
```

this is not so far from the true posterior mean

```
## [1] 0.8843
```

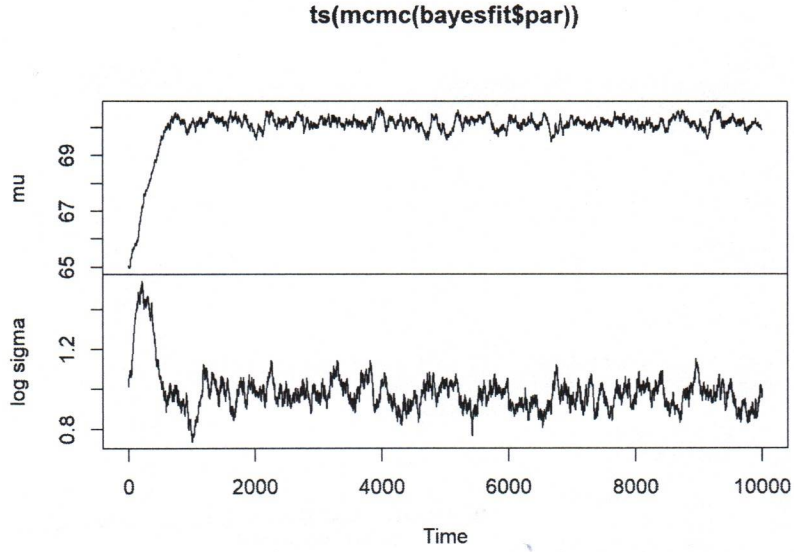
```
apply(bayesfit$par,2,mean)
```

```
## [1] 70.0077675  0.9835029
```

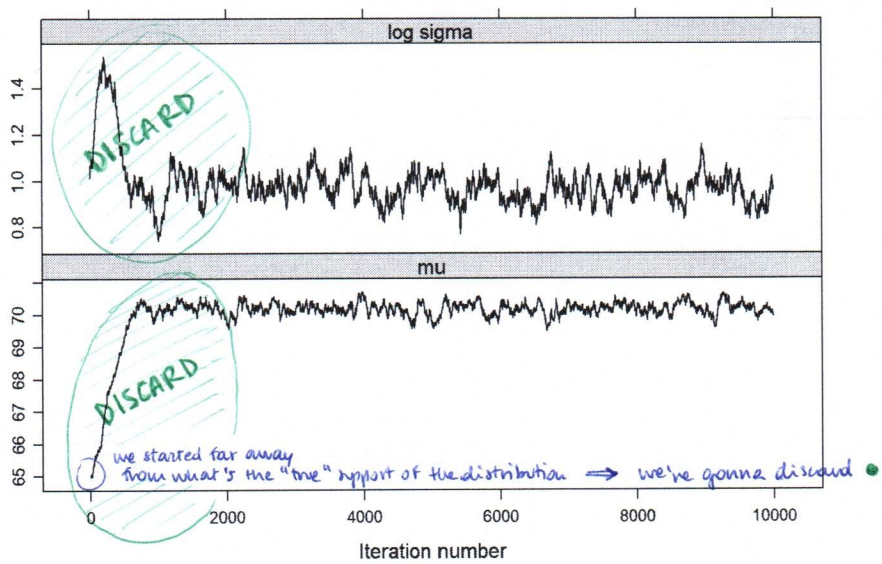
```
apply(bayesfit$par,2,sd)
```

```
## [1] 0.7339895 0.1037370
```

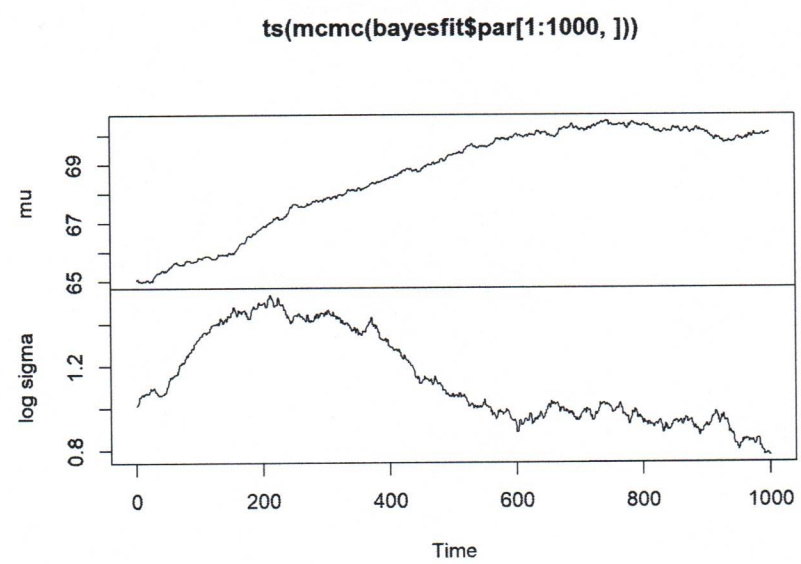
```
#Assegno i nomi alle colonne di bayesfit$par
dimnames(bayesfit$par)[[2]]=c("mu","log sigma")

# the key instruction in coda is "mcmc"
?mcmc
## See more details on R package CODA below!!

x11()
plot(ts(mcmc(bayesfit$par)))
```



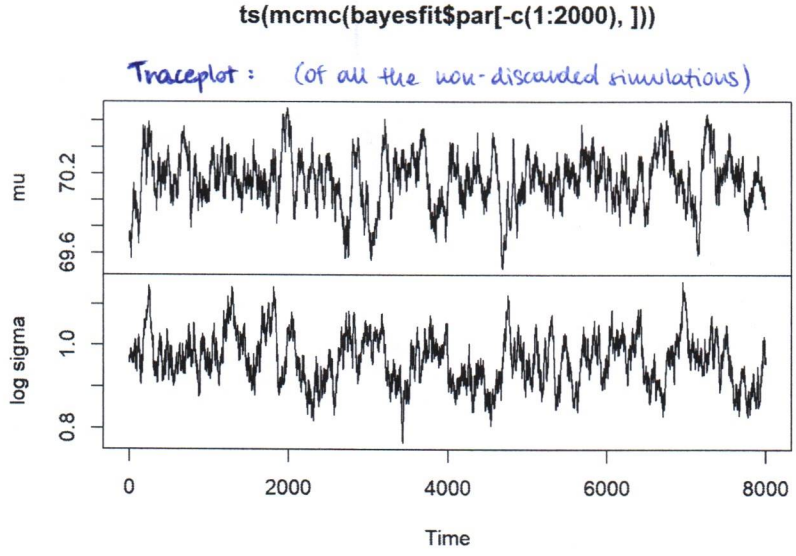

```
xyplot(mcmc(bayesfit$par),col="black") # "fancier" plot than above *
```




```
*
# "mcmc" (from coda) is used to create a Markov Chain Monte Carlo object
windows()
# BURN-IN: about 600 iterations
#xyplot(mcmc(bayesfit$par[1:1000,]),col="black")
plot(ts(mcmc(bayesfit$par[1:1000,])))
```



```
# If we forget the first 2000 iterations
#xyplot(mcmc(bayesfit$par[-c(1:2000),]),col="black")
plot(ts(mcmc(bayesfit$par[-c(1:2000),])))
```

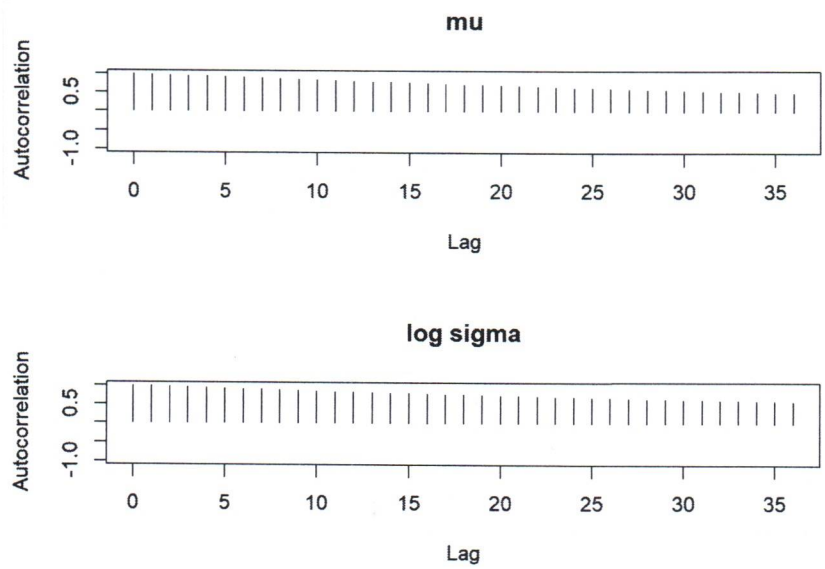


Traceplot: (of all the non-discarded simulations)

It's good, however we immediately see that the draws are highly autocorrelated (the correlation between one draw and the next one is high, we would prefer: ▒▒▒▒▒▒▒▒).

Because of this autocorrelation we're not exploring the support of the posterior distribution quickly enough.

```
# Sembra che già dopo le prime 2000 iterazioni si sia giunti a convergenza
# a 'good ' traceplot is a fat hairy caterpillar !
# a 'snake-like' caterpillar does NOT mean in general the the MC hasn't reached convergence,
# but we need to increase the sample size because there is high correlation

# For a given sample size, the ACCURACY of our inferences is dependent on the EFFICIENCY
# of our posterior samples. We'll see that the efficiency decreases with an increasing level
# of AUTOCORRELATION.

x11()
par(mfrow=c(2,1))
# Plots of AUTOCORRELATION between X_n and X_{n+k} by CODA
autocorr.plot(mcmc(bayesfit$par[-c(1:2000),]),auto.layout=FALSE)
```



Do we like it? Mmmh. They're not so bad, however the autocorrelation il pretty high.
⟹ we should either increase T (much larger T to control the variance of the estimator) or we can choose a better scale in the proposal

```
# Dall'HELP della funzione "autocorr": High autocorrelations within chains
# indicate slow mixing (the support of the target distribution has not been fully explored yet)
# and, usually, slow convergence.
# It may be useful to thin out a chain with high autocorrelations before calculating summary
# statistics: a thinned chain may contain most of the information, but take up
# less space in memory. Re-running the MCMC sampler with a different
# parameterization may help to reduce autocorrelation

summary(mcmc(bayesfit$par[-c(1:2000),]))
```

← computes the mean and the standard deviation of the marginal posterior distribution of each component (we're discarding the first 2000 iterations)

```
##
## Iterations = 1:8000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 8000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean      SD  Naive SE Time-series SE
## mu         70.1541 0.19681 0.0022004      0.023660
## log sigma  0.9662 0.06064 0.0006779      0.007829
##
## 2. Quantiles for each variable:
##
##              2.5%     25%     50%    75%  97.5%
## mu         69.7200 70.0331 70.1626 70.285 70.523
## log sigma  0.8548  0.9219  0.9667  1.009  1.087
```

square root of the posterior variance = MC error

somehow a "better estimator" than the Naive SE (for the same topic) = MC error

```
# We estimate the (square root of the) variance of the MCMC estimator, that is sigma^2_f / T
# col metodo delle batch means:
batchSE(mcmc(bayesfit$par[-c(1:2000),]), batchSize=50)
```

```
##          mu   log sigma
## 0.013669118 0.004305206
```

```
# Si ottiene un valore diverso da Naive SE e da Time-series SE !!

effectiveSize(mcmc(bayesfit$par[-c(1:2000),])) # su 8000 iterazioni
```

```
##        mu log sigma
##  69.19321  59.98847
```

```
# An estimate of the square root of sigma^2_f / T is called Monte Carlo standard error

# ------------------------------------------------------------------------------
# In order to check the convergence of the MC :
# ------------------------------------------------------------------------------
# - analysis of the traceplots (also to understand which burn-in we should consider).
# - MCerror/posterior sd has to be small (less than 0.1 o 1 o 5%);
#   l'MCerror viene stimato col metodo delle batch means o quello usato in "summary".
# - convergence diagnostics
# In order to check the mixing of the chain:
# - autocorrelation plot: the autocorrelation should be 'small' as the lag increases.
#   In fact, if there is a strong dependence between X_n and X_{n+k}, this means
#   that the value of X_{n+k} strongly depends on the previous value X_n, and therefore
#   the chain does not mix 'well', since the chain would be bound to stay in some region
#   of the support set, and not 'covering' the whole support of the target distribution.
# Slow mixing usually denotes a 'slow' convergence:
# one solution is to reparametrize the parameters in order to reduce autocorrelation.
# the other solution (more popular!) is to perform a process known as THINNING whereby only
# every kth value from the MC is actually stored for inference, so that autocorrelation among
# successive value of the chain is diminished.
# Note that this only represents an efficiency gain in terms of storing and post-processing
# the sample: for the same computational cost of simulation, the full sample size will always
# contain more information.

# Il numero di iterazioni da salvare (al meno del burnin e del thinning)
# per calcolare poi le medie ergodiche di stabilisce in base al MCerror
# che si vuole ottenere.
# L'Effective Sample Size (ESS) di una componente unidimensionale  di una MCMC è
# il numero di iterazioni indipendenti che dovrei effettuare dalla stessa distribuzione
# a posteriori per ottenere lo stesso MCerror; se l'ESS è piccolo, questo indica che la
# stima sarà 'povera', perchè la standard deviation del parametro sarà grande
# per tenere costante l'MCerror - più è grande l'ESS e meglio è...

# Ripeto l'algoritmo di M-H con un "migliore" punto iniziale e una migliore "scale"
start     = c(70,1)
proposal  = list(var=fit$var,scale=2.0)
bayesfit2 = rwmetrop(groupeddatapost,proposal,start,10000,d)

bayesfit2$accept
```

```
## [1] 0.2931
```

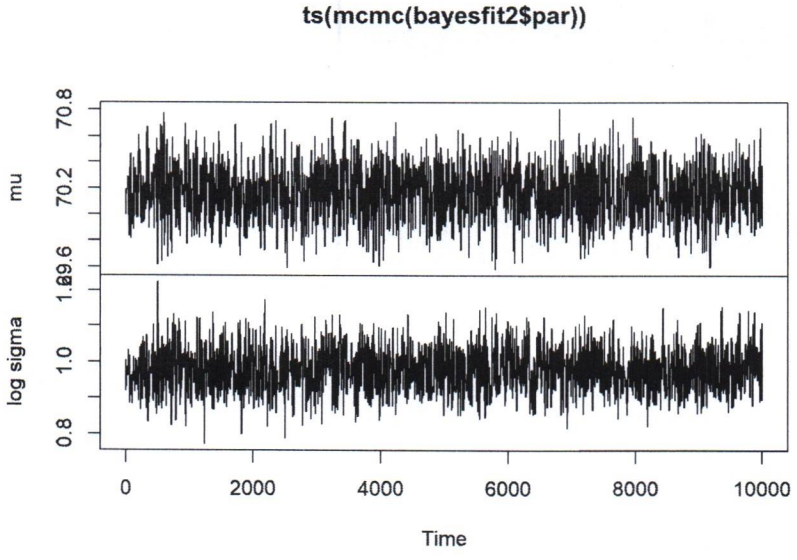
```
apply(bayesfit2$par,2,mean)
```

```
## [1] 70.1736161  0.9788022
```

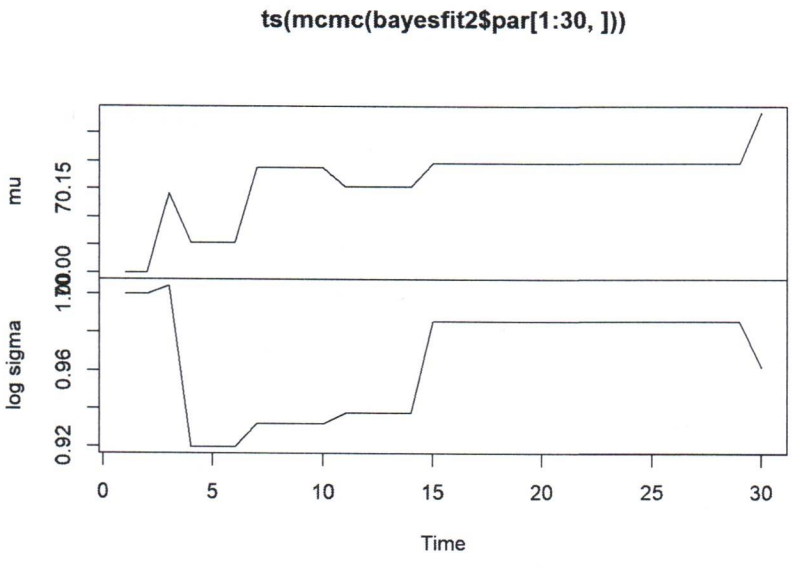
```
apply(bayesfit2$par,2,sd)
```

```
## [1] 0.18850198 0.05630485
```

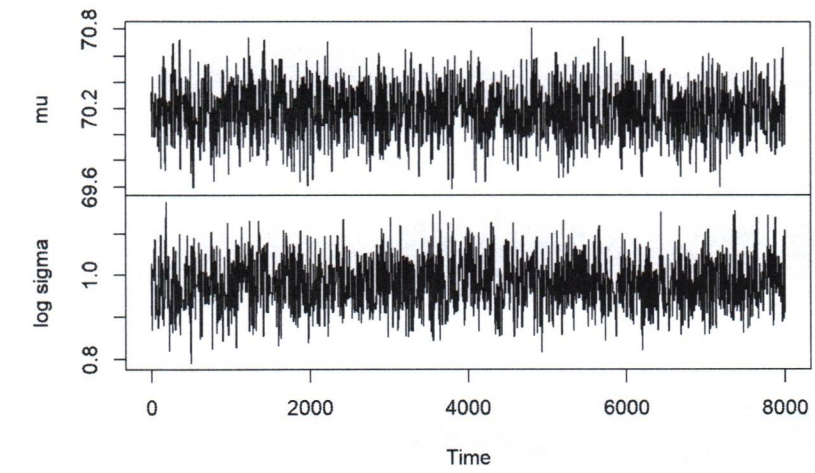
```
dimnames(bayesfit2$par)[[2]]=c("mu","log sigma")
#xyplot(mcmc(bayesfit2$par),col="black")
x11()
plot(ts(mcmc(bayesfit2$par)))
```



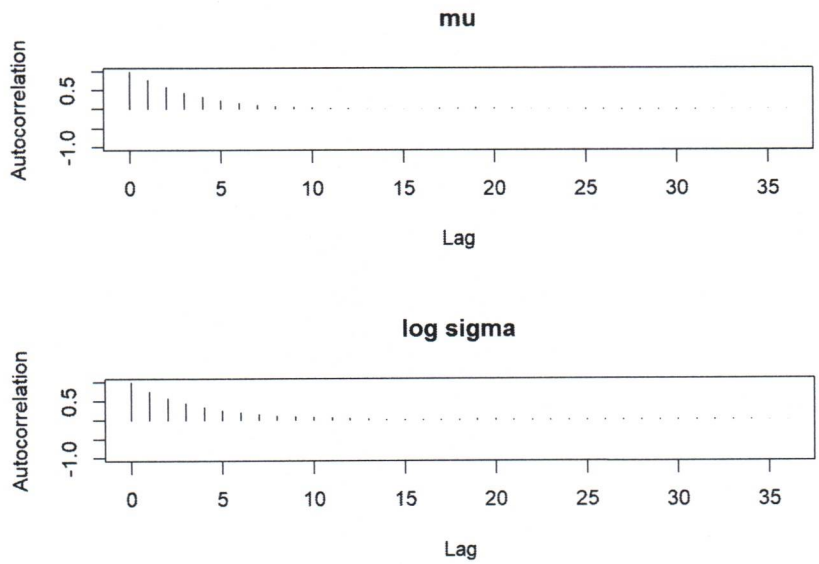

```
windows()
#xyplot(mcmc(bayesfit2$par[1:30,]),col="black")
plot(ts(mcmc(bayesfit2$par[1:30,])))
```




```
sim.parameters=mcmc(bayesfit2$par[-c(1:2000),])
#xyplot(mcmc(bayesfit2$par[-c(1:2000),]),col="black")
plot(ts(mcmc(bayesfit2$par[-c(1:2000),])))
```




```
x11()
par(mfrow=c(2,1))
autocorr.plot(sim.parameters,auto.layout=FALSE)
```



These autocorrelation plots are much better!

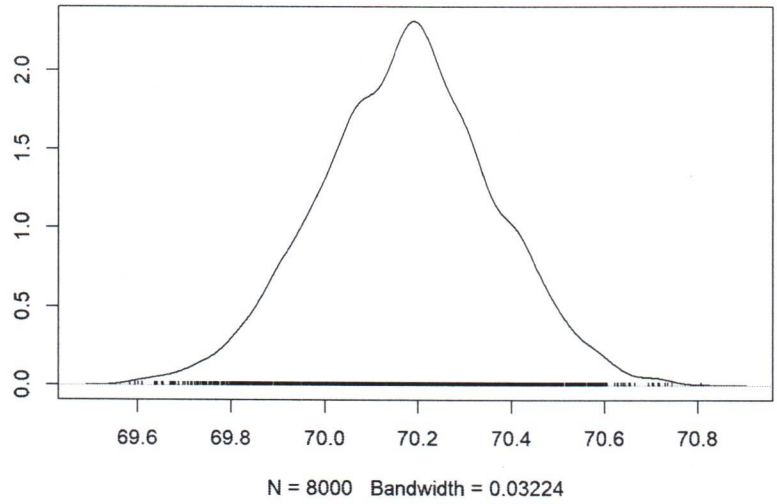
```
summary(sim.parameters)
```

```
##
## Iterations = 1:8000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 8000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean      SD  Naive SE Time-series SE
## mu        70.1742 0.18696 0.0020902       0.005741
## log sigma  0.9789 0.05573 0.0006231       0.001701
##
## 2. Quantiles for each variable:
##
##              2.5%    25%     50%    75%  97.5%
## mu        69.8060 70.053 70.1787 70.299 70.538
## log sigma  0.8767  0.939  0.9777  1.015  1.094
```

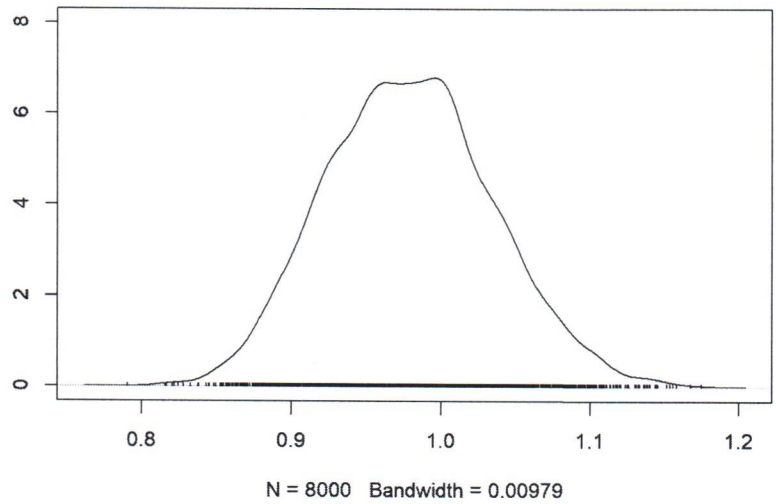
```
## The naive standard error is the standard error of the mean,
## which captures simulation error of the mean
batchSE(sim.parameters, batchSize=50)
```

```
##          mu   log sigma
## 0.005414502 0.001716500
```

```
# L'ultima colonna di "summary" e il comando batchSE stimano entrambi la radice quadrata
# di (sigma_f)^2/n che e' la varianza asintotica dell'errore Monte Carlo,
# ma batchSE è una stima MENO precisa della stima "time series"

x11()
#Kernel density delle posterior marginali
densplot(mcmc(bayesfit2$par[-c(1:2000),1]))
```




```
densplot(mcmc(bayesfit2$par[-c(1:2000),2]),ylim=c(0,8))
```




```
effectiveSize(mcmc(bayesfit2$par[-c(1:2000),]))
```

```
##        mu log sigma
##  1060.372  1073.000
```

much larger than before but still smaller than 8000

```
######## CODA package - updated version of a script by ILARIA BIANCHINI
######## Download the package from CRAN
######## See the manual!!!
##### First step: checking the convergence
# ****CODA package****
## The function `mcmc' is used to create a Markov Chain Monte Carlo
## object. The data are taken to be a vector, or a matrix with one
## column per variable.
niter  = 10000
burnin = 2000
thin   = 1
##########
th.post.mc <- mcmc(bayesfit$par[-c(1:2000),],
                   start = burnin + 1,
                   end = niter, thin = thin)
dim(th.post.mc)
```

```
## [1] 8000    2
```

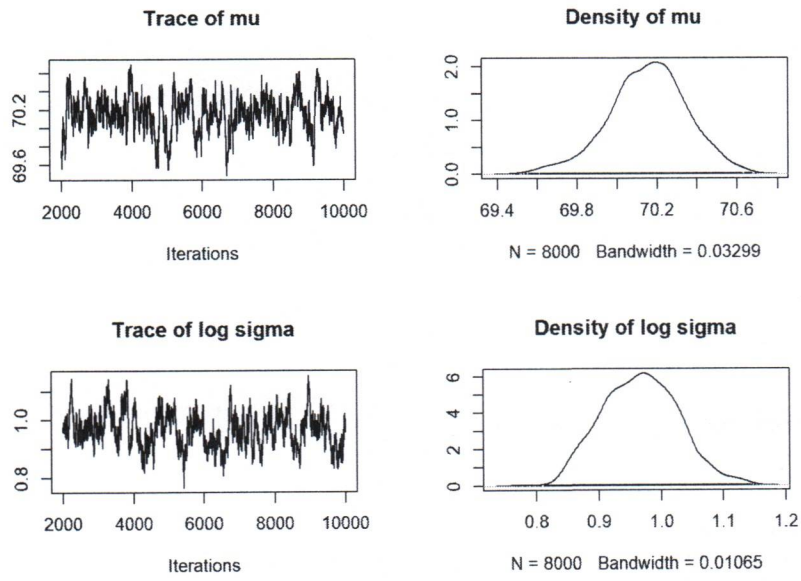
```
is(th.post.mc)
```

```
## [1] "mcmc"
```

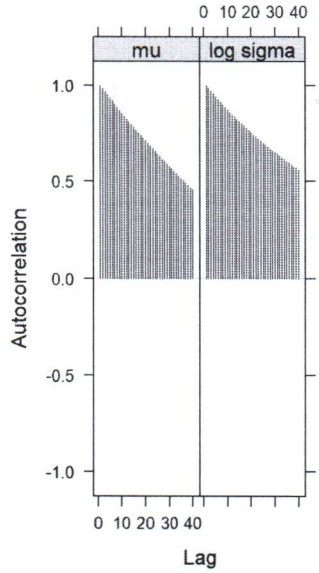
```
## Parameters  start, end, thin are info we give to "mcmc".
## We are not asking to take a subset of th.post
## But we are giving information on how we got th.post

## We can do summary() of an mcmc object to get summary statistics
## for the posterior.
summary(th.post.mc)
```

```
##
## Iterations = 2001:10000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 8000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean      SD  Naive SE Time-series SE
## mu        70.1541 0.19681 0.0022004       0.023660
## log sigma  0.9662 0.06064 0.0006779       0.007829
##
## 2. Quantiles for each variable:
##
##              2.5%     25%     50%    75%  97.5%
## mu        69.7200 70.0331 70.1626 70.285 70.523
## log sigma  0.8548  0.9219  0.9667  1.009  1.087
```

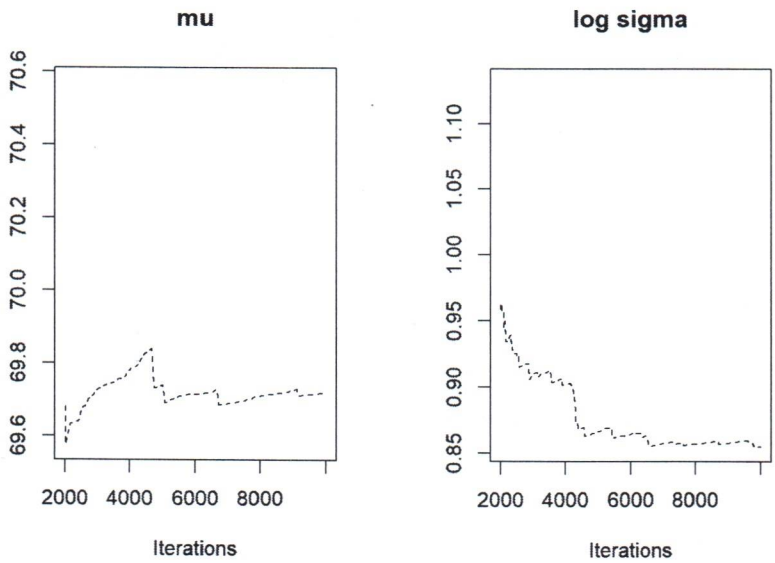
```
x11()
plot(th.post.mc)
```




```
##  The naive standard error is the standard error of the mean,
##  which captures simulation error of the mean rather than posterior
##  uncertainty.
##
##  naiveSE = posteriorSD/sqrt(n)
##
##  The time-series standard error adjusts the naive standard error for
##  autocorrelation. ---> more precise estimate

# Funzioni di autocovarianza
# Autocorrelation of the chain:
x11()
acfplot(th.post.mc)
```



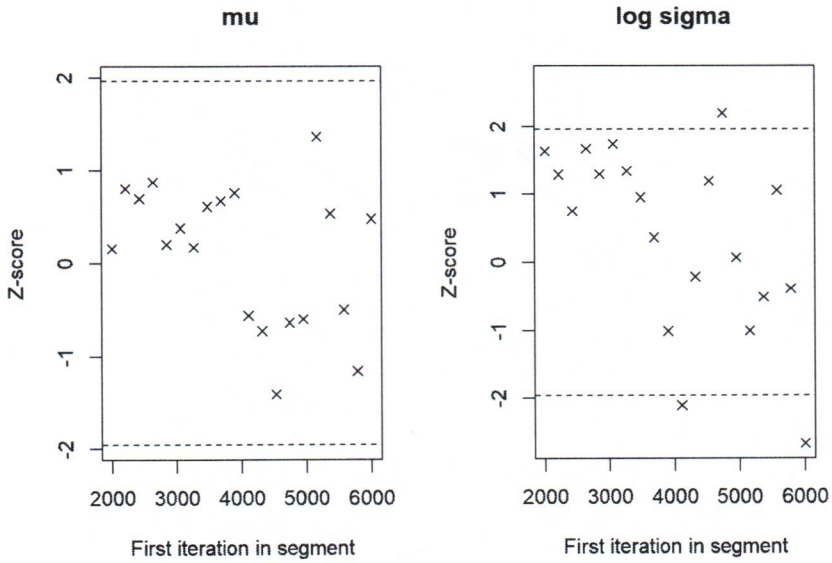

```
# Cumulative plot:
cumuplot(th.post.mc)
```




```
# in cumuplot, the dashed lines correspond to the quantiles
# 2.5% and 97.5%. The thick central lines represents the median of the chain
# up to the i-th iteration i=burnin+1...niter

#?geweke.diag
# Geweke (1992) proposed a convergence diagnostic for Markov chains
# based on a test for equality of the means of the first and last
# part of the MCMC (by default the first 10% and the last 50%).
# If the samples were drawn from the stationary distribution
# of the chain, the two means are equal and Geweke's statistic has
# an asymptotically standard normal distribution.

# Geweke diagnostic
geweke.diag(th.post.mc)
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##        mu log sigma
##    0.1624    1.6380
```

```
# the command returns Z-scores for a test of equality of means
# between the first and last parts of the chain. A separate
# statistic is calculated for each variable in each chain.
# Note: If the chain has converged, most of these z-scores (one for each par)
#     should fall within 2 standard deviations from zero

# Geweke-Brooks plot
# If 'geweke.diag' indicates that the first and last part of a
# sample from a Markov chain are not drawn from the same
# distribution, it may be useful to discard the first few iterations
# to see if the rest of the chain has "converged". This plot shows
# what happens to Geweke's Z-score when successively larger numbers
# of iterations are discarded from the beginning of the chain. To
# preserve the asymptotic conditions required for Geweke's
# diagnostic, the plot never discards more than half the chain.
# The first half of the Markov chain is divided into 'nbins - 1'
# segments, then Geweke's Z-score is repeatedly calculated. The
# first Z-score is calculated with all iterations in the chain, the
# second after discarding the first segment, the third after
# discarding the first two segments, and so on. The last Z-score is
# calculated using only the samples in the second half of the chain.

x11()
geweke.plot(th.post.mc, frac1 = 0.1, frac2 = 0.5, nbins = 20)
```



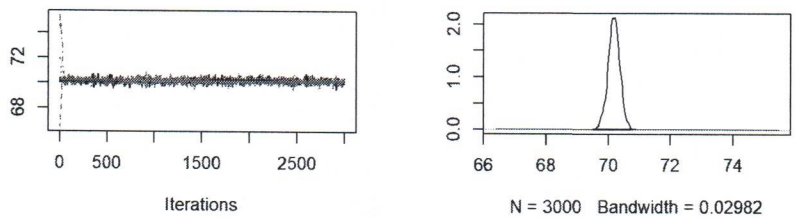

```
# In both the diagnostics we used the effective sample size
# ?effectiveSize
# This Effective Sample Size gives an estimate
# of the equivalent number of independent iterations that the chain represents.
# effectiveSize(th.post.mc)
# It is computed in this way:
#            Neff = n/(1+2\sum_k rho_k)
# where \rho_k is the autocorrelation function at lag k.
effectiveSize(th.post.mc)
```

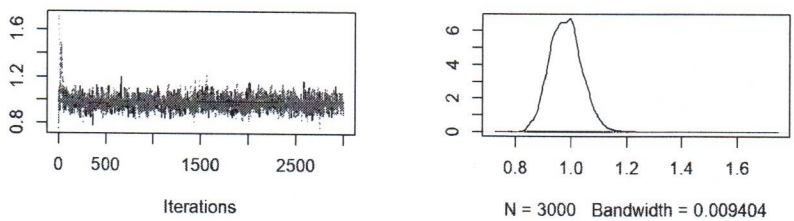
```
##        mu log sigma
##  69.19321  59.98847
```

```
dim(th.post.mc)[1]
```

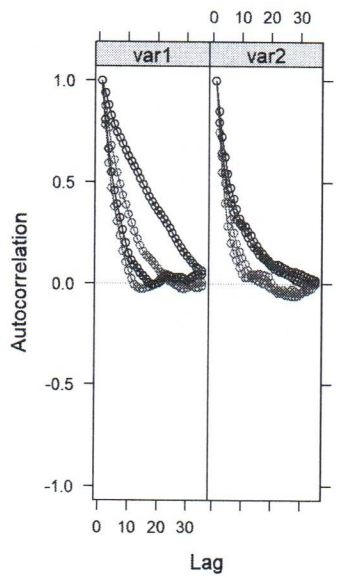
```
## [1] 8000
```

```
# So in 8000 correlated iterations, it is as if I had
# Only 65 (or 81) independent samples!
# This is because we use a Markov Chain Monte Carlo algorithm,
# which produces not independent samples

### Methods based on more than one chain
# Idea: we start several chains with different starting
#       points;  if the convergence is reached, the trajectories
#       should be similar

# Start 10 chains with different starting points:
start=c(70,1)
proposal=list(var=fit$var,scale=2.0)
th.post1 <- mcmc(rwmetrop(groupeddatapost,proposal,start,3000,d)$par)

start=c(67,0.6)
proposal=list(var=fit$var,scale=2.0)
th.post2 <- mcmc(rwmetrop(groupeddatapost,proposal,start,3000,d)$par)

start=c(72,1.7)
#proposal=list(var=fit$var,scale=2.0)
th.post3 <- mcmc(rwmetrop(groupeddatapost,proposal,start,3000,d)$par)

start=c(75,0.75)
#proposal=list(var=fit$var,scale=2.0)
th.post4 <- mcmc(rwmetrop(groupeddatapost,proposal,start,3000,d)$par)

# Put everything in a list
quattro.mc <- mcmc.list(th.post1, th.post2, th.post3, th.post4)
# Draw all the chains:
x11()
plot(quattro.mc )
```





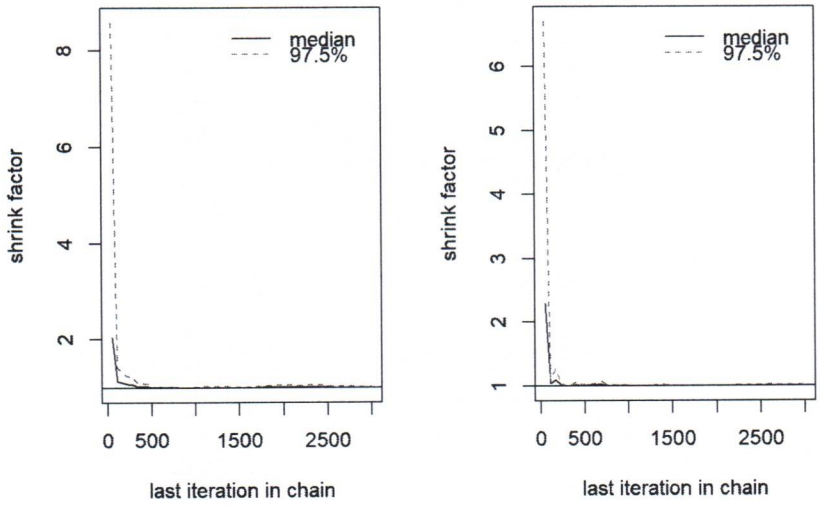

```
acfplot(quattro.mc)
```



```
# Gelman and Rubin statistics ************
# Steps (for each parameter):
# 1. Run m>= 2 chains of length 2n from overdispersed starting values.
# 2. Discard the first n draws in each chain.
# 3. Calculate the within-chain and between-chain variance.
# 4. Calculate the estimated variance of the parameter as a
#    weighted sum of the within-chain and between-chain variance.
# 5. Calculate the potential scale reduction factor.

# Note: here W, the within chain variance, is the mean of the
#  variances of each chain and B, the between chain variance, is
#  the variance of the chain means multiplied by n.

# We can estimate the variance of the stationary distribution as
# a weighted average of W and B.
# The potential scale reduction factor is given by
#      R = sqrt(Var_hat(theta)/W)
# and is defined for each parameter.
# When R is high (greater than 1.2) then we should run our chains
# out longer to improve convergence to the stationary distribution
gelman.diag(quattro.mc)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1       1.02
## [2,]          1       1.01
##
## Multivariate psrf
##
## 1.01
```

```
# ?gelman.plot
# This plot shows the evolution of Gelman and Rubin's shrink factor
# as the number of iterations increases.
x11()
gelman.plot(quattro.mc)
```



Example: (Chp. 6 Hoff's book – MIDGE DATA)

y_i = length of a wing of the i-th midge (in mm)

$n = 9$

A PRIORI:

$\mathbb{E}[\theta] = 1.9$ $\quad$ $\mathrm{Var}(\theta) = (0.95)^2$

$\mathbb{E}[\tau^2] = \frac{1}{\sigma^2} = 100$ $\quad$ $\mathrm{Var}(\tau^2) = \frac{2}{\nu_0 (\sigma_0^2)^2} = 20.000$

Model that we're going to use:

$$\begin{cases} Y_1, \dots, Y_n \mid \theta, \sigma^2 \overset{iid}{\sim} N\left(\theta, \frac{1}{\tau^2}\right), \quad \sigma^2 = \frac{1}{\tau^2} \\ \theta \sim N(\mu_0, t_0^2) \\ \tau^2 \sim \text{gamma}\left(\frac{\nu_0}{2}, \frac{\nu_0 \sigma_0^2}{2}\right) \end{cases} \quad \text{a priori}$$

A POSTERIORI:

$$\pi(\theta, \tau^2 \mid \underline{y}) \propto \underbrace{\left[\prod_{i=1}^{n} (\tau^2)^{\frac{1}{2}} e^{-\tau^2 \frac{(y_i-\theta)^2}{2}}\right]}_{\text{likelihood}} \cdot \underbrace{\left[e^{-\frac{(\theta-\mu)^2}{2/t_0^2}}\right]}_{\text{prior for } \theta} \cdot \underbrace{\left[(\tau^2)^{\frac{\nu_0}{2}-1} e^{-\frac{\nu_0\sigma_0^2}{2}\tau^2} \mathbb{1}_{(0,\infty)}(\tau^2)\right]}_{\text{prior for } \tau^2}$$

(we're dropping the constants)

$$\ast\ast \longrightarrow \propto (\tau^2)^{\frac{n}{2}} e^{-\tau^2 \sum_{i=1}^{n} \frac{(y_i-\theta)^2}{2}} e^{-\frac{(\theta-\mu_0)^2}{2t_0^2}} (\tau^2)^{\frac{\nu_0}{2}-1} e^{-\frac{\nu_0\sigma_0^2}{2}\tau^2} \mathbb{1}_{(0,\infty)}(\tau^2)$$

$$= \chi(\underline{Y}, \theta, \tau^2) = \chi(\underline{Y} \mid \theta, \tau^2) \cdot \chi(\theta, \tau^2)$$

what we need to do is design a Gibbs sampler with the target distribution that is the posterior distribution:

target distribution: $\chi(\theta, \tau^2 \mid \underline{Y})$

we need to find the two:

FULL-CONDITIONALS:

- $[\theta \mid \tau^2, \underline{Y}]$ * $\quad$ $(/\ \chi(\theta \mid \tau^2, \underline{Y}) \ /\ \chi(\theta \mid \text{rest}))$
- $[\tau^2 \mid \theta, \underline{Y}]$ * $\quad$ $(/\ \chi(\tau^2 \mid \theta, \underline{Y}) \ /\ \chi(\tau^2 \mid \text{rest}))$

(the squared brackets stay for "full conditionals")

* How can we compute this full-conditional? We have to find the joint distribution of the 3 components $(\theta, \tau^2, \underline{Y})$ and divide by the marginal distribution of τ^2 and $\underline{Y}$.
OTHERWISE we can consider **, which is the joint distribution of data and both parameters and we focus only on the factors that have θ in them.
⟹ The first full-conditional is simply ** but when we ignore everything that does not have θ (we keep only the factors that have θ)

* We want to use the same approach. We consider ** and we take only the factors that contain τ

- $[\theta \mid \tau^2, \underline{Y}] \propto e^{-\tau^2 \sum_{i=1}^{n} \frac{(y_i-\theta)^2}{2} - \frac{(\theta-\mu_0)^2}{2t_0^2}}$ ← here we have to recognize the model (kernel) with some manipulations (because we need to sample from it)

$$\propto e^{-\frac{1}{2}(n\tau^2 + \frac{1}{t_0^2})(\theta-\theta_n)^2}$$

$$\theta_n = \frac{n\bar{y}\tau^2 + \frac{\mu_0}{t_0^2}}{n\tau^2 + \frac{1}{t_0^2}}$$

$$\sim \boxed{N\left(\theta_n[\tau^2], \frac{1}{(n\tau^2 + \frac{1}{t_0^2})}\right)}$$

here we mean that θ_n depends on τ^2

- $[\tau^2 \mid \theta, \underline{Y}] \propto (\tau^2)^{\frac{n+\nu_0}{2}-1} e^{-\tau^2\left[\frac{\sum_{i=1}^{n}(y_i-\theta)^2}{2} + \frac{\nu_0\sigma_0^2}{2}\right]} \mathbb{1}_{(0,\infty)}(\tau^2)$

$$\sim \boxed{\text{gamma}\left(\frac{n+\nu_0}{2}, \frac{\sum_{i=1}^{n}(y_i-\theta)^2}{2} + \frac{\nu_0\sigma_0^2}{2}\right)}$$

⟹ we're able, using R, to sample from these 2 full-conditionals (normal and gamma)

⟹ we've able to sample the Gibbs sampler starting from any initial point (in this case bivariate: initial point for θ and initial point for τ^2): we can sample alternatively from the first full conditional and then from the second full conditional.
The $\underline{Y}$ will be always fixed, since it's the data, but what do we fix as τ^2 in $[\theta|\tau^2, \underline{Y}]$? and what as θ in $[\tau^2|\theta, \underline{Y}]$? The last available value that we have simulated so far.

```
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
### Chapter 6 - Hoff's book
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
# The data are wing lenght (in mm) of n=9 midges (moscerini)
# The model : Y_i are conditionally iid N(theta,sigma^2)
# Both theta and tau=1/sigma^2 are unknown
# Prior: pi(theta,tau)= pi(theta)* pi(tau)
#        theta ~ N(mu0, t20) (mu0 is the mean, t20 is the variance)
#        tau ~ gamma(nu0/2, (nu0*s20)/2 )

mu0 = 1.9
t20 = 0.95^2
s20 = .01
nu0 = 1

1/s20
```

```
## [1] 100
```

```
# data
y      = c(1.64,1.70,1.72,1.74,1.82,1.82,1.82,1.90,2.08)
n      = length(y)
mean.y = mean(y)
var.y  = var(y)

# Generate a Gibbs sampler MCMC of S draws
set.seed(1)
S   <- 1000
PHI <- matrix(nrow=S,ncol=2) # Inizializzo una matrice con 2 colonne e S righe
                             # it contains simulated values of the BIVARIATE Gibbs sampler MC
PHI[1,]<-phi<-c( mean.y, 1/var.y) # Initial point of the chain
### ---------------------------------------------------------------------------
### Gibbs sampling
### ---------------------------------------------------------------------------
for(s in 2:S) {
  # the bidim. vector phi contains the "current" value of the chain
  # generate a new theta value from its full conditional
  mun    <- ( mu0/t20 + n*mean.y*phi[2] ) / ( 1/t20 + n*phi[2] ) #updated mean
  t2n    <- 1/( 1/t20 + n*phi[2]                                ) #updated variance
  phi[1] <- rnorm(1, mun, sqrt(t2n) )

  # generate a new tau=1/sigma^2 value from its full conditional
  nun    <- nu0+n                              # it does NOT change with the iteration
  s2n    <- (nu0*s20 + (n-1)*var.y + n*(mean.y-phi[1])^2)/nun   #updated parameter
  phi[2] <- rgamma(1, nun/2, nun*s2n/2)
  PHI[s,] <- phi
}

PHI
```

Gibbs

for θ we assume the mean of the empirical data (because θ represent the conditional expectation of the data)
for τ^2 we act similarly and as the initial value we set the reciprocal of the empirical variance

this is what we defined θ_n

here instead of θ we use the last generated θ, which is : phi[1]

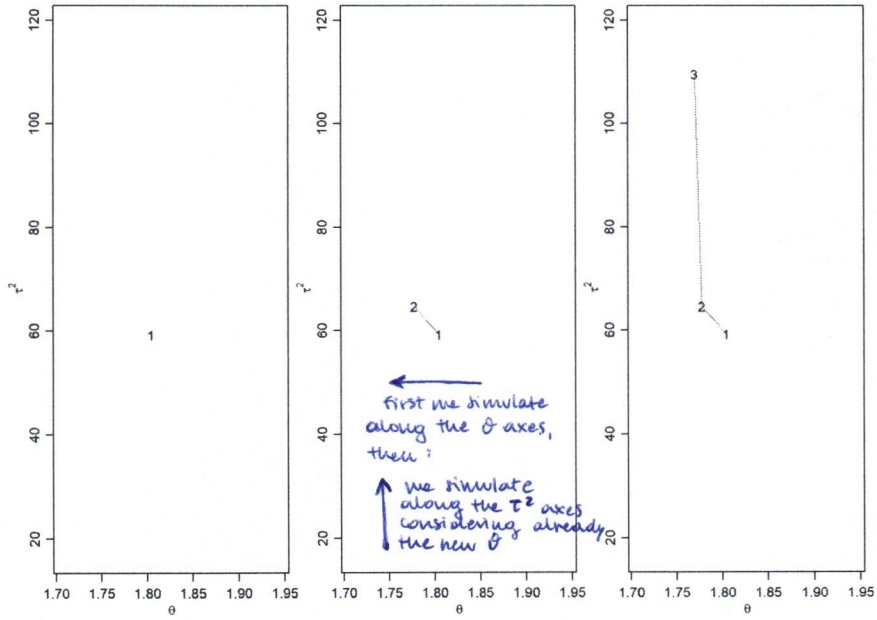
Notice that we're using:

- **phi** :  which contains the latest simulations of θ and τ^2
- **PHI** :  which contains all the simulations of θ and τ^2

```
##             [,1]       [,2]
##    [1,] 1.804444  59.249506
##    [2,] 1.777542  64.644769
##    [3,] 1.770015 109.452135
##    [4,] 1.815045  40.097871
##    [5,] 1.789247  61.016227
##    [6,] 1.791618 113.012706
##    [7,] 1.816766  44.793608
##    [8,] 1.789824  49.942034
##    [9,] 1.803917  92.741227
##   [10,] 1.832977  76.771355
##   [11,] 1.839530  80.871704
##   [12,] 1.807351  17.496111
##   [13,] ..
```

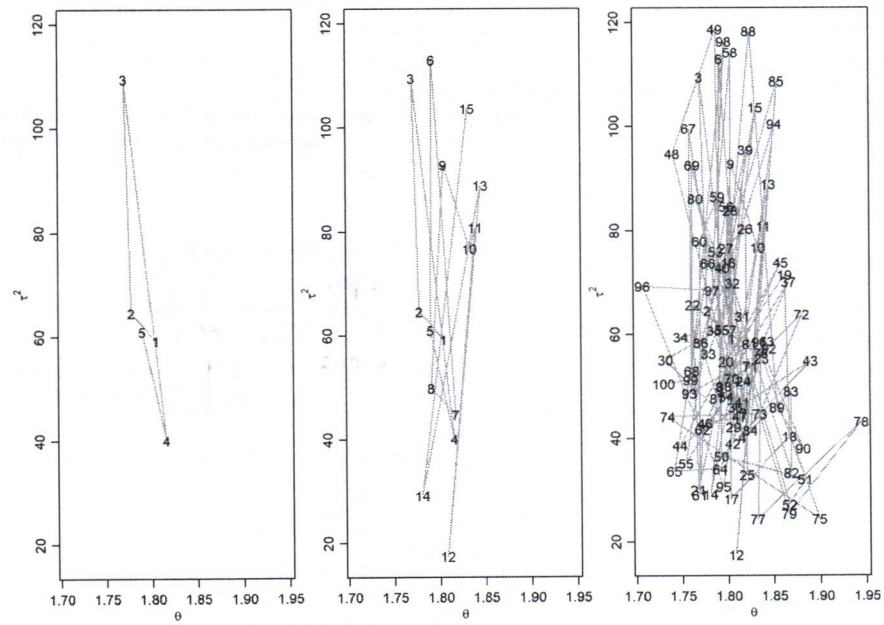
```
### ---------------------------------------------------------------------------
### First iterations of Gibbs sampler
### ---------------------------------------------------------------------------
x11()
par(mfrow=c(1,3),mar=c(2.75,2.75,.5,.5),mgp=c(1.70,.70,0))
m1<-1
plot(PHI[1:m1,],type="p",xlim=range(PHI[1:100,1]), ylim=range(PHI[1:100,2]),
     lty=1,col="gray",xlab=expression(theta),ylab=expression(tau^2))
text(PHI[1:m1,1], PHI[1:m1,2], c(1:m1))

m1<-2
plot(PHI[1:m1,],type="l",xlim=range(PHI[1:100,1]), ylim=range(PHI[1:100,2]),
     lty=1,col="gray",xlab=expression(theta),ylab=expression(tau^2))
text(PHI[1:m1,1], PHI[1:m1,2], c(1:m1))

m1<-3
plot(PHI[1:m1,],type="l",xlim=range(PHI[1:100,1]), ylim=range(PHI[1:100,2]),
     lty=1,col="gray",xlab=expression(theta),ylab=expression(tau^2))
text(PHI[1:m1,1], PHI[1:m1,2], c(1:m1) )
```

We've simulating from a bidimensional posterior $\Longrightarrow$ $(\theta, \tau^2) \in \mathbb{R}^2$

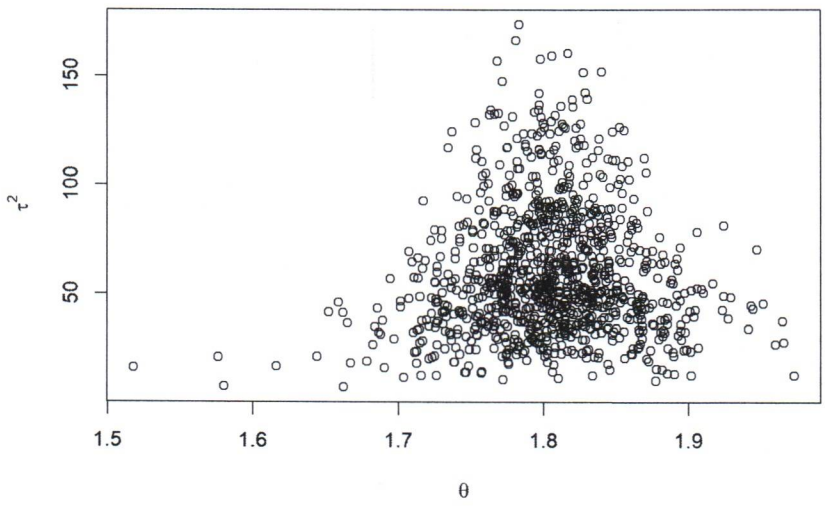


first 3 iterations of the method

```
par(mfrow=c(1,3),mar=c(2.75,2.75,.5,.5),mgp=c(1.70,.70,0))
m1<-5
plot(PHI[1:m1,],type="l",xlim=range(PHI[1:100,1]), ylim=range(PHI[1:100,2]),
     lty=1,col="gray",xlab=expression(theta),ylab=expression(tau^2))
text(PHI[1:m1,1], PHI[1:m1,2], c(1:m1))

m1<-15
plot(PHI[1:m1,],type="l",xlim=range(PHI[1:100,1]), ylim=range(PHI[1:100,2]),
     lty=1,col="gray",xlab=expression(theta),ylab=expression(tau^2))
text(PHI[1:m1,1], PHI[1:m1,2], c(1:m1))

m1<-100
plot(PHI[1:m1,],type="l",xlim=range(PHI[1:100,1]), ylim=range(PHI[1:100,2]),
     lty=1,col="gray",xlab=expression(theta),ylab=expression(tau^2))
text(PHI[1:m1,1], PHI[1:m1,2], c(1:m1))
```




```
sseq <- 1:1000

x11()
plot(PHI[sseq,1],PHI[sseq,2],xlab=expression(theta), ylab=expression(tau^2) ,
     xlim=range(PHI[,1]),ylim=range(PHI[,2]))
```

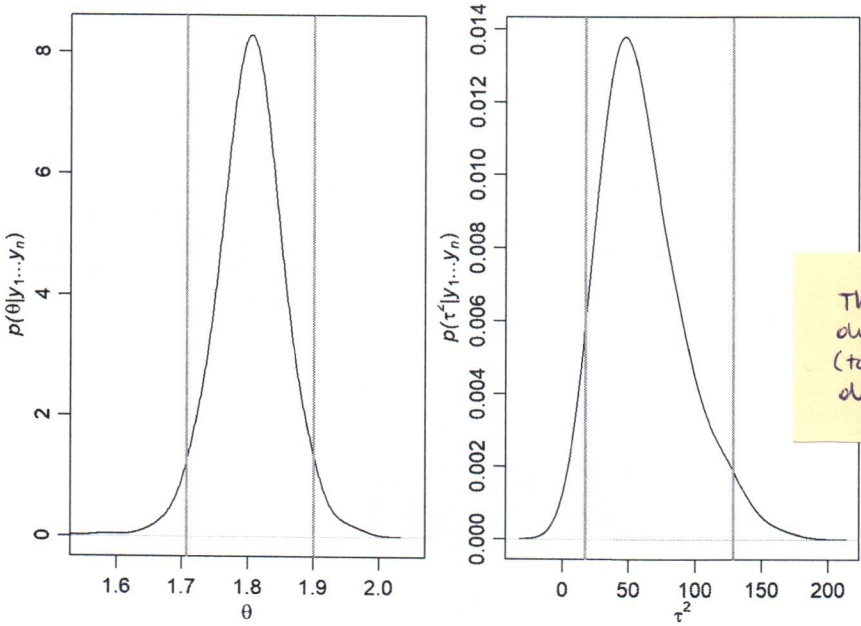


(same as before but without labels)

```
### ---------------------------------------------------------------------------
### PLOTS of the posterior marginal densities, with credible intervals
### ---------------------------------------------------------------------------
x11()
par(mfrow=c(1,2),mar=c(2.75,2.75,.5,.5),mgp=c(1.70,.70,0))
sseq<-1:1000

# image(mean.grid,prec.grid,post.grid,col=gray( (10:0)/10 ),
#       xlab=expression(theta), ylab=expression(tilde(sigma)^2),
#       xlim=range(PHI[,1]),ylim=range(PHI[,2]) )

plot(density(PHI[,1],adj=2), xlab=expression(theta),main="", xlim=c(1.55,2.05),
     ylab=expression( paste(italic("p("),
     theta,"|",italic(y[1]),"...",italic(y[n]),")",sep="")))
abline(v=quantile(PHI[,1],prob=c(.025,.975)),lwd=2,col="gray")

plot(density(PHI[,2],adj=2), xlab=expression(tau^2),main="",
     ylab=expression( paste(italic("p("),
     tau^2,"|",italic(y[1]),"...",italic(y[n]),")",sep="")))
abline(v=quantile(PHI[,2],prob=c(.025,.975)),lwd=2,col="gray")
```

this is to estimate the kernel density for θ given the data (marginal distribution)

→ for the credible interval



We plot the kernel density estimator of the raw data.

← We have the two marginal distributions (estimated) and the credible intervals (95% credible interval)

Those are the tipical plots used during the projects!
(to show what is the posterior distribution of the data)

```
quantile(PHI[,1],c(.025,.5,.975))          # Posterior CI of theta
```

```
##     2.5%      50%    97.5%
## 1.707282 1.804348 1.901129
```

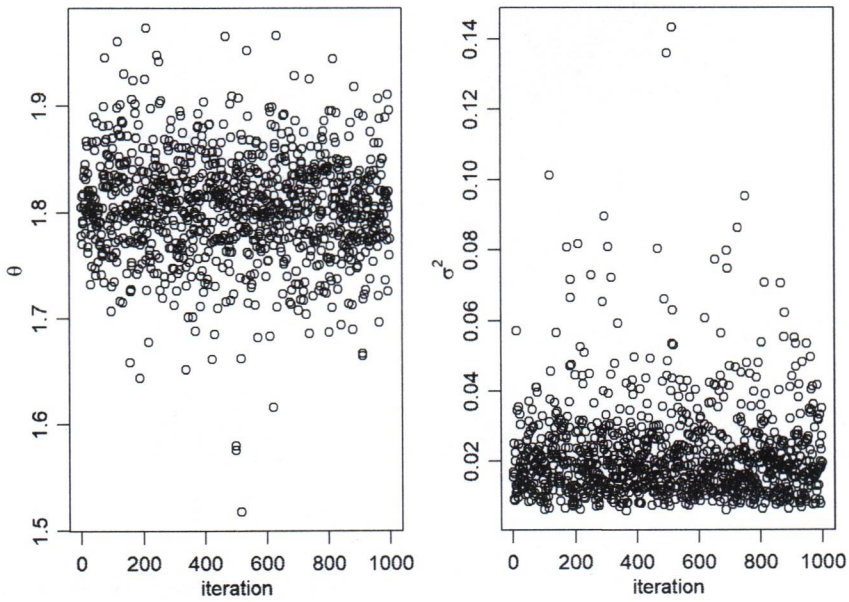
```
quantile(PHI[,2],c(.025,.5, .975))          # Posterior CI of tau
```

```
##      2.5%       50%     97.5%
##  17.48020  53.62511 129.20020
```

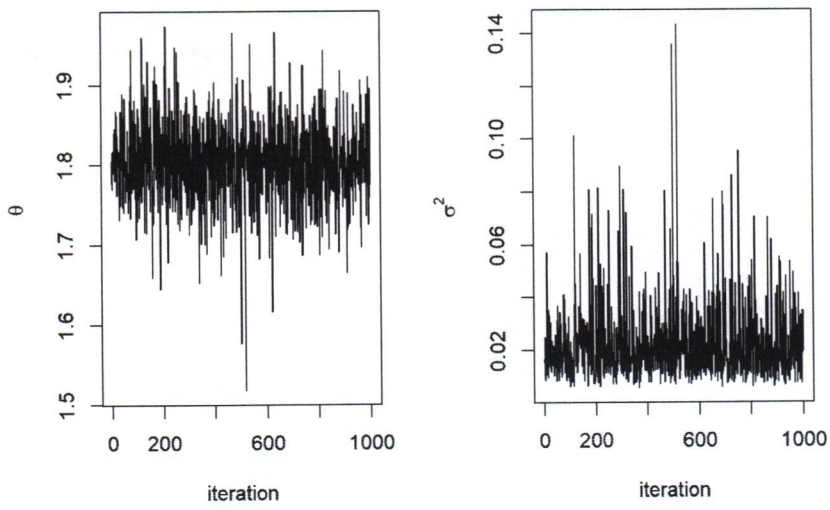
```
quantile(1/sqrt(PHI[,2]),c(.025,.5, .975))  # Posterior CI of sigma
```

```
##       2.5%        50%      97.5%
## 0.08797701 0.13655763 0.23918408
```

```
### --------------------------------------------------------------------------------
### CONVERGENCE DIAGNOSTICS
### --------------------------------------------------------------------------------
#pdf("fig6_7.pdf",family="Times",height=3.5,width=7)
x11()
par(mfrow=c(1,2))
par(mar=c(3,3,1,1),mgp=c(1.75,.75,0))
plot(PHI[,1],xlab="iteration",ylab=expression(theta))
plot(1/PHI[,2],xlab="iteration",ylab=expression(sigma^2))
```



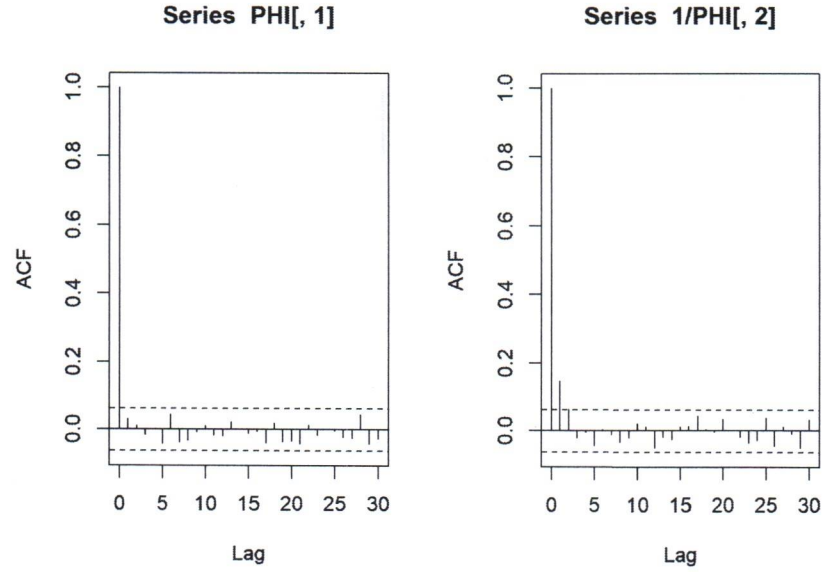

```
### --------------------------------------------------------------------------------
### Marginal traceplots
### --------------------------------------------------------------------------------
x11()
par(mfrow=c(1,2))
plot(ts(PHI[,1]),xlab="iteration",ylab=expression(theta))
plot(ts(1/PHI[,2]),xlab="iteration",ylab=expression(sigma^2))

library(coda)
```



We can see that there is no need for burn-in. This is because we started close to stationarity for both parameters.

```
par(mfrow=c(1,2))
acf(PHI[,1]) -> tmp1
acf(1/PHI[,2]) -> tmp2
```

Autocorrelation:



Notice: these are too good. In reality we shouldn't expect something like this.

```
# The Effective Sample Size (ESS) of a one-dimensional component of a MCMC is
# the number of INDEPENDENT iterations to achieve the same MCerror from
# the same marginal distribution of the MC (the stationary dist).
# For a time series x of length N, the standard error of the mean is var(x)/n where n is
# the effective sample size. n = N only when there is no autocorrelation.
# If the ESS is small (compared to the sample size of the chain), this means that the
# MC estimate will be rather 'poor', because the variance of the MC estimate will be 'Large'
# wrt the variance of the Monte Carlo estimator.
# The larger the ESS, the better!

effectiveSize( PHI[,1] )
```

```
## var1
## 1000
```

$\Longrightarrow \sum_{j=1}^{\infty} \hat{\rho}_j = 0$

```
effectiveSize(1/PHI[,2] )
```

```
##     var1
## 742.6481
```

→ smaller but still good

EFFECTIVE SAMPLE SIZE = number of ≠ draws (we want it as high as we can)

The linear model

Alessandra Guglielmi

Politecnico di Milano
Dipartimento di Matematica
Milano, Italia
e-mail: alessandra.guglielmi@polimi.it

~~28 November 2016~~ 2020

Homoscedastic linear models

Suppose that we have the data (y_i, x_i) $i = 1, \dots, n$ where y_i are the responses and x_i are the covariates (p-dim). We suppose y_i to be continuous. We're interested in finding correlation between $\mathbb{E}[Y_i]$ and x_i. We represent:

$Y_i = x_i^T \beta + \varepsilon_i$

$\varepsilon_i \overset{iid}{\sim} N(0, \sigma^2)$ eq. to: $Y_i \overset{iid}{\sim} N(x_i^T \beta, \sigma^2)$

$\boldsymbol{Y}$ vector of continuous responses, dim=n
X $n \times p$-matrix, with $p = k + 1$, its rows are the vectors $\boldsymbol{x}_i = (1, x_{i1}, \dots, x_{ik})^T$ covariates of subject i in the sample

For a few slides: $n > p$ and $rank(X) = p$ ($\Leftrightarrow X^T X$ invertible)

The parameters are:
$\beta = (\beta_0, \beta_1, \dots \beta_k)^T$, $\sigma^2 > 0$; $\tau := 1/\sigma^2$ (τ is the precision)

We assume

$$\boldsymbol{Y} | X, \beta, \sigma^2 \sim \mathcal{N}_n(X\beta, \sigma^2 I_n)$$

Parameters of interest: (β, σ^2) or (β, τ)

Likelihood

$$L(\beta, \sigma^2; \boldsymbol{y}) \propto \frac{1}{(\sigma^2)^{n/2}} e^{-\frac{(\boldsymbol{y}-X\beta)^T(\boldsymbol{y}-X\beta)}{2\sigma^2}}$$

or, equivalently,

$$L(\beta, \tau; \boldsymbol{y}) \propto \tau^{n/2} e^{-\frac{\tau}{2}(\boldsymbol{y}-X\beta)^T(\boldsymbol{y}-X\beta)}$$

Frequentist estimate

MLE of β: $\hat{\beta} = (X^T X)^{-1} X^T \boldsymbol{y}$ $\quad \mathrm{Var}(\hat{\beta} | \sigma^2) = \sigma^2 (X^T X)^{-1}$

unbiased estimator for σ^2:

$$\hat{\sigma}^2 = \frac{1}{n-p}(\boldsymbol{y} - X\hat{\beta})^T(\boldsymbol{y} - X\hat{\beta}) = \frac{s^2}{n-p},$$

where s^2 is the sum of squared residuals

estimate of the covariance matrix of β: $\hat{\sigma}^2 (X^T X)^{-1} = \hat{\sigma}^2 [\omega_{ij}]$

t-statistics:

$$T_i = \frac{\hat{\beta}_i - \beta_i}{\sqrt{\hat{\sigma}^2 \omega_{ii}}} \sim t(n-p)$$

We want to see what are the conjugate priors for this model when σ^2 is known and when σ^2 is unknown (together with β). To do this we have to re-write the likelihood. How can we rewrite $(y - X\beta)^T(y - X\beta)$?

More on the likelihood

$$(\boldsymbol{y} - X\beta)^T(\boldsymbol{y} - X\beta) = (\boldsymbol{y} - X\hat{\beta})^T(\boldsymbol{y} - X\hat{\beta}) + (\beta - \hat{\beta})^T X^T X(\beta - \hat{\beta})$$
$$= s^2 + (\beta - \hat{\beta})^T X^T X(\beta - \hat{\beta})$$

residuals squares

so that:

$$L(\beta, \sigma^2; \boldsymbol{y}) \propto \frac{1}{(\sigma^2)^{n/2}} e^{-\frac{s^2}{2\sigma^2} - \frac{1}{2\sigma^2}(\beta-\hat{\beta})^T X^T X(\beta-\hat{\beta})}$$

or, equivalently,

$$L(\beta, \tau; \boldsymbol{y}) \propto \tau^{n/2} e^{-\frac{\tau s^2}{2} - \frac{\tau}{2}(\beta-\hat{\beta})^T X^T X(\beta-\hat{\beta})}$$

$$(y - X\beta)^T(y - X\beta) =$$
$$= (y - X\hat{\beta} + X\hat{\beta} - X\beta)^T(y - X\hat{\beta} + X\hat{\beta} - X\beta)$$
$$= (y - X\hat{\beta})^T(y - X\hat{\beta}) + (X\hat{\beta} - X\beta)^T(X\hat{\beta} - X\beta) +$$
$$+ 2(X\hat{\beta} - X\beta)^T(y - X\hat{\beta})$$
$$= s^2 + (\hat{\beta} - \beta)^T X^T X(\hat{\beta} - \beta) +$$
$$+ 2(\hat{\beta} - \beta)^T X^T(y - X\beta)$$

since $X^T y - X^T X\hat{\beta} = 0$

$$= s^2 + (\beta - \hat{\beta})^T X^T X(\beta - \hat{\beta})$$

The covariates are fixed or random?

Y_i is our data and we assume a model for our data (gaussian distributed).
X_i is the information about something and we should consider the informations as random variables and assume a model also for them

There two options are equivalent.

1. fixed covariates (as in standard frequentist analysis)
2. random covariate, but we assume that, a priori, X and (β, σ^2) are independent:

$$f(\boldsymbol{y}, X|\omega, \beta, \sigma^2) = f(\boldsymbol{y}|X, \omega, \beta, \sigma^2)p(X|\omega, \beta, \sigma^2)$$
$$= f(\boldsymbol{y}|X, \beta, \sigma^2)p(X|\omega);$$

hence, since the interest here is in (β, σ^2), $p(X|, \omega)$ represents a constant factor in the likelihood and therefore it can be discarded.

the covariates are realizations of random variables s.t. the model for $\underline{X}$ is $\perp\!\!\!\perp$ on (β, σ^2) (parameters which specify the distribution of $\underline{Y}$)

(the distribution of $\underline{X}$ is $\perp\!\!\!\perp$ of the parameters of interest: β, σ^2)

σ^2 KNOWN

Conjugate prior for β when σ^2 is known

Going back to the likelihood (p.5): if we know σ^2, a good prior for β should be a gaussian distribution (as a function of β the likelihood corresponds to a kernel of the gaussian distr.).

We assume a prior that:
$\beta \sim \mathcal{N}_p(\boldsymbol{b}_0, B_0)$, B_0 invertible $p \times p$-matrix; then the posterior is such that

$$\beta|\boldsymbol{y} \propto \exp\{-\frac{\tau}{2}(\beta - \hat{\beta})^T X^T X(\beta - \hat{\beta}) - \frac{1}{2}(\beta - \hat{\boldsymbol{b}}_0)^T B_0^{-1}(\beta - \hat{\boldsymbol{b}}_0)\}$$

We make calculations similar to the case without covariates ($p = 1$):

$$\beta|\boldsymbol{y} \sim \mathcal{N}_p((\tau X^T X + B_0^{-1})^{-1}(\tau X^T X\hat{\beta} + B_0^{-1}\boldsymbol{b}_0), (\tau X^T X + B_0^{-1})^{-1})$$

the posterior mean of β is a weighted average of the prior mean $\boldsymbol{b}_0$ and the MLE $\hat{\beta}$

(the weights are $\tau X^T X$ and B_0^{-1})

σ^2 UNKNOWN

Conjugate prior

We do not assume that $X^T X$ is invertible!

In the expression $L(\beta, \sigma^2; \boldsymbol{y})$ of the likelihood, β (as a function of σ^2) occurs as in the kernel of a Gaussian distribution, and $\frac{1}{\sigma^2}$ as in the kernel of a gamma distribution

Conjugate prior:

$$\pi(\beta, \sigma^2) = \pi(\beta|\sigma^2)\pi(\sigma^2),$$

with

PRIOR
$$\begin{cases} \beta|\sigma^2 \sim \mathcal{N}_p(\boldsymbol{b}_0, \sigma^2 B_0) \\ \sigma^2 \sim inv - gamma\left(\frac{\nu_0}{2}, \frac{\nu_0\sigma_0^2}{2}\right), \end{cases}$$

where $\boldsymbol{b}_0 \in \mathbb{R}^p$, B_0 invertible $p \times p$-matrix, and $\nu_0, \sigma_0^2 > 0$

Conjugate prior: the posterior distribution

Calculations yield:

$$\pi(\beta, \sigma^2|\boldsymbol{y}) = \pi(\beta|\sigma^2, \boldsymbol{y})\pi(\sigma^2|\boldsymbol{y}),$$

where

POSTERIOR

$$\begin{cases} \beta|\sigma^2, \boldsymbol{y}, X \sim \mathcal{N}_p(\boldsymbol{b}_n, \sigma^2 B_n) \\ \sigma^2|\boldsymbol{y}, X \sim inv-gamma\left(\frac{\nu_n}{2}, \frac{\nu_n \sigma_n^2}{2}\right), \end{cases}$$

with $B_n = (X^T X + B_0^{-1})^{-1}$, $\boldsymbol{b}_n$ is the convex linear combination of $\hat{\beta}$ and $\boldsymbol{b}_0$:

$$\boldsymbol{b}_n = B_n(X^T X\hat{\beta} + B_0^{-1}\boldsymbol{b}_0) = (X^T X + B_0^{-1})^{-1}(X^T\boldsymbol{y} + B_0^{-1}\boldsymbol{b}_0)$$

$$\nu_n = \nu_0 + n,$$

$$\sigma_n^2 = \frac{1}{\nu_n}\left(\nu_0\sigma_0^2 + s^2 + (\boldsymbol{b}_0 - \hat{\beta})^T(B_0 + (X^T X)^{-1})^{-1}(\boldsymbol{b}_0 - \hat{\beta})\right)$$

$$= \frac{1}{\nu_n}\left(\nu_0\sigma_0^2 + \boldsymbol{b}_0^T B_0^{-1}\boldsymbol{b}_0 + \boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{b}_n^T B_n^{-1}\boldsymbol{b}_n\right)$$

Conjugate prior: the marginal posterior of β

$$\beta|\boldsymbol{y}, X \sim t_p(\boldsymbol{b}_n, \sigma_n^2 B_n, \nu_n)$$

By definition $\boldsymbol{Y} = (Y_1, \ldots, Y_q)^T \sim t_q(\mu, \Sigma, \nu)$ when

$$f_{\boldsymbol{Y}}(\boldsymbol{y}) = \frac{\Gamma(\frac{q+\nu}{2})}{\Gamma(\frac{\nu}{2})}\frac{1}{\sqrt{(det\Sigma)(\nu\pi)^q}}\left(1 + \frac{1}{\nu}(\boldsymbol{y} - \mu)^T\Sigma^{-1}(\boldsymbol{y} - \mu)\right)^{-\frac{q+\nu}{2}}$$

multivariate t distribution

If $\nu > 1$: $\mathbb{E}(\boldsymbol{Y}) = \mu$
If $\nu > 2$: $\mathrm{Var}(\boldsymbol{Y}) = \frac{\nu}{\nu-2}\Sigma$

Generalization of the one-dimensional t distribution $t_1(\mu, \sigma^2, \nu)$, with location μ, scale σ, degrees of freedom ν

Conjugate prior: the predictive distribution of $\boldsymbol{Y}_{new}$

How can we **predict** the response of a new individual coming with its vector of covariates ?

$\boldsymbol{Y}_{new}$ is a vector of new observations, corresponding to a new $m \times p$ covariate matrix X_{new}

$$X^{new} = \begin{bmatrix} (\underline{x}_1^{new})^T \\ \vdots \\ (\underline{x}_m^{new})^T \end{bmatrix} \Big\} m \text{ rows}$$

p columns

$$\boldsymbol{Y}_{new} = X_{new}\beta + \epsilon_{new}, \text{ with } \epsilon_{new} \sim \mathcal{N}_m(0, \sigma^2 I_m),$$

and $\epsilon_{new}, (\epsilon_1, \ldots, \epsilon_n)$ conditionally independent.

Integrating out the likelihood in $\boldsymbol{Y}_{new}$ wrt the posterior, we get:

$$\boldsymbol{Y}_{new}|\boldsymbol{y}, X, X_{new} \sim t_m\left(X_{new}\boldsymbol{b}_n, \sigma_n^2(I_m + X_{new}B_n X_{new}^T), \nu_n\right)$$

Analogy with unidim Gaussian model with unknown mean and variance - covariates

We assumed Y_i's iid $\mathcal{N}(\mu, \sigma^2)$, conditionally to μ, σ^2, i.e.

$$\boldsymbol{Y}|\mu, \sigma^2 \sim \mathcal{N}_n(\boldsymbol{\mu}, \sigma^2 I_n), \quad \boldsymbol{\mu} = (\mu, \ldots, \mu)$$

Conjugate prior: $\mu|\sigma^2 \sim \mathcal{N}(\mu_0, \frac{\sigma^2}{n_0})$, $\sigma^2 \sim inv-gamma(\frac{\nu_0}{2}, \frac{\nu_0\sigma_0^2}{2})$

Posterior:

$$\mu|\sigma^2, \boldsymbol{y} \sim \mathcal{N}(\mu_1, \frac{\sigma^2}{n_1}) \quad \sigma^2|\boldsymbol{y} \sim inv-gamma(\frac{\nu_1}{2}, \frac{\nu_1\sigma_1^2}{2})$$

where

$$\mu_1 = \frac{n_0\mu_0 + n\bar{y}}{n_0 + n}, \quad n_1 = n_0 + n, \quad \nu_1 = \nu_0 + n,$$

$$\sigma_1^2 = \frac{\frac{n_0 n}{n_0+n}(\mu_0 - \bar{y})^2 + (n-1)s^2 + \nu_0\sigma_0^2}{\nu_0 + n}$$

Alternative priors for the parameters:

Zellner's g prior

How to choose hyperparameters in the prior?
The most difficult aspect is the derivation of B_0.

Zellner's g prior:

PRIOR

$$\begin{cases} \beta|\sigma^2, X \sim \mathcal{N}_p(\boldsymbol{b}_0, \sigma^2 B_0) \text{ con } B_0 = c\,(X^TX)^{-1} \\ \sigma^2 \sim inv-gamma\left(\frac{0}{2}, \frac{0 \times \sigma_0^2}{2}\right) \propto \frac{1}{\sigma^2}\mathbf{1}_{(0,+\infty)}(\sigma^2), \end{cases}$$

We need to assume X^TX invertible!

By definition, Zellner's g prior is the conjugate prior under $B_0 = c\,(X^TX)^{-1}$ and $\nu_0 = 0$.
REMARK: it depends on X, but not on the data (we assume the model is given conditioning wrt X)

Zellner's g prior: the meaning of c

We find that

$$\mathbb{E}(\beta|\boldsymbol{y}, X) = \frac{1}{c+1}\boldsymbol{b}_0 + \frac{c}{c+1}\hat{\beta}$$

→ the posterior expectation of β is a linear convex combination of b_0 and $\hat{\beta}$. (where $\hat{\beta}$ are MLE)

If $c = 1$: same weight on the prior information and on the sample, since in this case,

$$\mathbb{E}(\beta|\boldsymbol{y}, X) = \frac{1}{2}\boldsymbol{b}_0 + \frac{1}{2}\hat{\beta}$$

When $c = 100$: the prior gets a weight corresponding to 1% of the sample, since

$$\mathbb{E}(\beta|\boldsymbol{y}, X) = \frac{1}{101}\boldsymbol{b}_0 + \frac{100}{101}\hat{\beta} \approx \frac{1}{100}\boldsymbol{b}_0 + \frac{99}{100}\hat{\beta}$$

⇒ we're giving weight to the empirical component

! Rule of thumb: $c \approx \log(n)$ (n = sample size)

Zellner's g prior: the posterior

Apply the formula for the general posterior in the conjugate case when $B_0 = c(X^TX)^{-1}$ and $\nu_0 = 0$:

POSTERIOR

$$\begin{cases} \beta|\sigma^2, \boldsymbol{y}, X \sim \mathcal{N}_p(\boldsymbol{b}_n, \sigma^2 B_n) \\ \sigma^2|\boldsymbol{y}, X \sim inv-gamma\left(\frac{n}{2}, \frac{1}{2}(s^2 + (\hat{\beta} - \boldsymbol{b}_0)^T B_n^{-1}(\hat{\beta} - \boldsymbol{b}_0))\right), \end{cases}$$

where

$$B_n = (X^TX + B_0^{-1})^{-1} = \frac{c}{c+1}(X^TX)^{-1},$$

$$\boldsymbol{b}_n = B_n(X^TX\hat{\beta} + B_0^{-1}\boldsymbol{b}_0) = \frac{1}{c+1}\boldsymbol{b}_0 + \frac{c}{c+1}\hat{\beta}$$

An alternative (3rd) prior:

Reference prior: the posterior posterior

Prior:
$\pi(\beta, \sigma^2) \propto \frac{1}{\sigma^2}\mathbf{1}_{(0,+\infty)}(\sigma^2)$, i.e. $\pi(\beta, \tau) \propto \frac{1}{\tau}\mathbf{1}_{(0,+\infty)}(\tau)$

The posterior is the product of the following distributions:

$$\beta|\tau, \boldsymbol{y}, X \sim \mathcal{N}_p(\hat{\beta}, \frac{1}{\tau}(X^TX)^{-1})$$

$$\tau|\boldsymbol{y}, X \sim gamma(\frac{n-p}{2}, \frac{s^2}{2})$$

The posterior is proper if X^TX is invertible and $n > p$.
Equivalently:

$$\beta|\sigma^2, \boldsymbol{y}, X \sim \mathcal{N}_p(\hat{\beta}, \sigma^2(X^TX)^{-1})$$

$$\sigma^2|\boldsymbol{y}, X \sim inv-gamma(\frac{n-p}{2}, \frac{s^2}{2})$$

Reference prior: the predictive distribution of $\boldsymbol{Y}_{new}$

As before, let $\boldsymbol{Y}_{new}$ be a vector of q new observations, corresponding to a new $m \times p$ covariate matrix X_{new}

Recall that $\boldsymbol{Y}_{new}$ and $\boldsymbol{Y}$, conditionally to β, σ^2, are independent, so that integrating out the likelihood in $\boldsymbol{Y}_{new}$ wrt the posterior, we get:

$$\boldsymbol{Y}_{new}|\boldsymbol{y}, X, X_{new} \sim t_m\left(X_{new}\hat{\beta}, \frac{s^2}{n-p}(I_m + X_{new}(X^TX)^{-1}X_{new}^T), n-p\right)$$