Introduction au logiciel JAGS

Anne Philippe

Laboratoire de Mathématiques Jean Leray
Université de Nantes

March 18, 2015

http://www.math.sciences.univ-nantes.fr/∼philippe/
Anne.Philippe@univ-nantes.fr

Utilisation de Jags avec la libraire rjags de R

> *JAGS est un logiciel qui permet d'approximer la loi a posteriori d'un modèle bayésien via des algorithmes MCMC.*

En entrée : :

1. le modèle Vraisemblance + loi a priori
2. les données

En sortie :

1. une ou plusieurs réalisations d'une chaine de Markov de loi invariante la loi a posteriori

Définition d'un DAG

Un graphe orienté acyclique (DAG) $\mathcal{G}$ est formé par

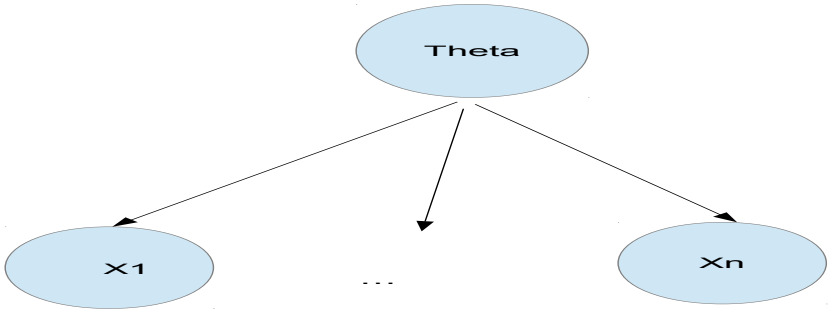
- un ensemble de sommets
- un ensemble d'arêtes dirigées qui ne constitue pas de boucle.

1. Les sommets représentent les variables du modèles
   - Observations
   - Paramètres
2. Pour tout sommet g du graphe $\mathcal{G}$, conditionnellement à ses parents $par(g)$, g est indépendant des autres sommets à l'exception de ses descendants $ch(g)$

$$p(\mathcal{G}) = \prod_{g \in \mathcal{G}} p(g|par(g))$$

DAG d'un modèle bayesien

- On dispose de n observations $X_1 ... X_n$ n iid suivant P_θ
- $\theta \sim \pi$ (le paramètre)



- $par(\theta) = \emptyset$
- $par(\mathbf{x}_i) = \theta$ pour tout $i = 1, ..., n$

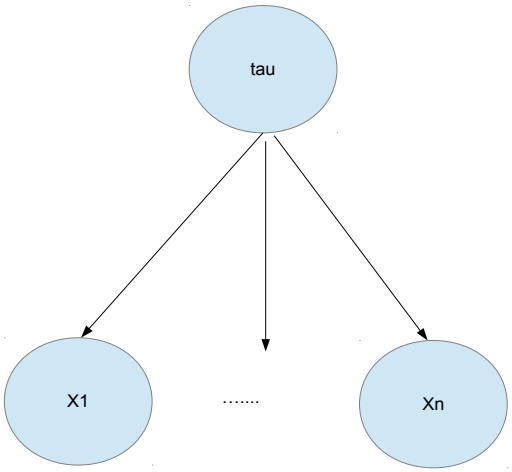
La loi jointe de $(x_1, ..., x_n, \theta)$ est donnée par

$$g(x_1, ..., x_n, \theta) = \pi(\theta) \prod_{i=1}^{n} f(x_i|\theta)$$

Exemple du modèle de Poisson

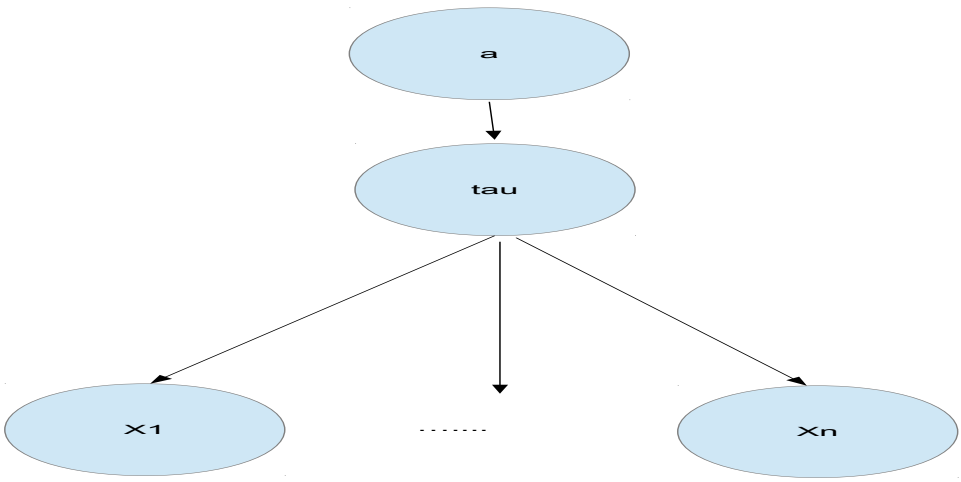
Observations iid : $X_i \sim Poisson(\tau) \qquad i = 1, ..., n$

1. Choix de la loi a priori : $\tau \sim Exponential(a)$ avec a fixé
2. La loi a posteriori de τ est la loi Gamma $(\sum X_i + 1, n + a)$
3. DAG du modèle :



modèle hiérarchique pour le modèle de Poisson

1. Choix de la loi a priori :
   $\tau \sim Exponential(a)$
   $a \sim Exponential(1)$
2. DAG du model



Traduction du DAG en langage BUGS/JAGS

Modèle avec *a* fixé :

```
model
{
  for( i in 1 : N ) {
    x[i] ~ dpois(tau)
    }

  tau ~ dgamma(1,a)
}
```

Modèle hiérarchique

```
model
{
  for( i in 1 : N ) {
    x[i] ~ dpois(tau)
    }

  tau ~ dgamma(1,a)
     a~ dgamma(1,1)
}
```

Ce code doit être stocké dans un fichier ici on stocke le code du modèle hiérarchique dans le fichier `modelPoisson.R`

Exécution de JAGS à partir de R via la librarie rjags

compilation

```
library(rjags)
#nombre de données
N <- 10
#x contient les données
#ici simulées suivant la loi de poisson de paramètre 2
x <- rpois(N, 2)
jags <- jags.model('modelPoisson.R',
                   data = list('x' = x,'N' = N),
                   n.chains = 1,
                   n.adapt = 1000)
```

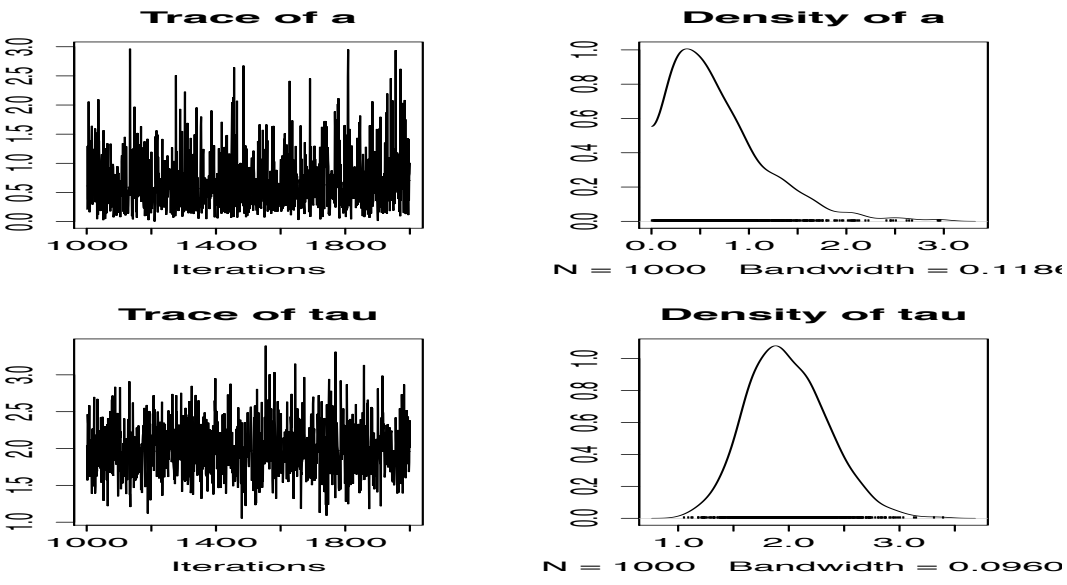
- On va simuler une chaine [`n.chains=1`].
- `n.adapts` est le nombre d'itérations pour calibrer les paramètres de l'algorithme d'Hasting Metropolis

Simulation et stockage des chaînes de Markov

```
# période de "chauffe"

> update(jags, 1000)
  |**************************************************| 100%

# samp contient la chaine de markov (10000 itérations)
# pour les paramètres tau et a

>samp = coda.samples(jags,c('tau','a'),10000)
 |**************************************************| 100%
```

Exploitation des résultats

```
> plot(samp)
```



Statistiques élémentaires

```
> summary(samp)
Iterations = 1001:2000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
      Mean     SD Naive SE Time-series SE
a   0.6838 0.4901   0.0155        0.01753
tau 1.9863 0.3606   0.0114        0.01225
2. Quantiles for each variable:
      2.5%    25%    50%    75% 97.5%
a   0.0815 0.3212 0.5659 0.9179 1.955
tau 1.3617 1.7277 1.9559 2.2155 2.730
```

Approximation des régions HPD

```
> HPDinterval(samp)
[[1]]
          lower    upper
a   0.04157356 1.652573
tau 1.37755688 2.739469
attr(,"Probability")
[1] 0.95
```

Critère de Gelman et Rubin

- Il faut simuler plusieurs chaines indépendantes
- On execute la fonction jags.model avec le paramètre n.chain > 1

```
> gelman.diag(samp)
Potential scale reduction factors:

    Point est. Upper C.I.
a            1          1
tau          1          1

Multivariate psrf

1
```

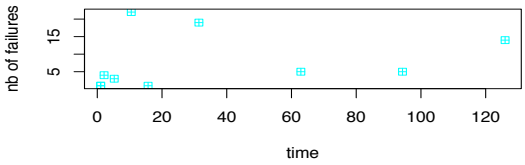
limite de JAGS : loi a priori impropre

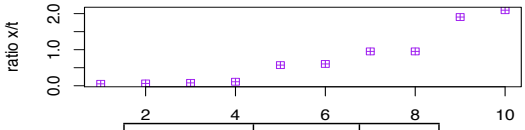
On ne peut pas définir des modèles avec des lois a priori impropres
Une alternative est de remplacer les lois impropres par des lois de probabilité ayant une grande variance ...
Par exemple

- pour la mesure de Lebesgue : on peut prendre une loi gaussienne centrée avec une très grande variance
  (dnorm(0,10^{-6}) le second paramètre est la précision)
- pour la loi $\pi(\sigma^2) = 1/\sigma^2$: on peut prendre une loi inverse gamma avec des paramètres proches de zéro.
  $\tau \sim$ dgamma(0.001,0.001) and $\sigma = 1/\sqrt{\tau}$

Application : Reliability of 10 power plant pumps

The number of failures X_i is assumed to follow a Poisson distribution





$$X_i \sim Poisson(\theta_i * t_i) \qquad i = 1, ..., 10$$

where

- θ_i is the failure rate for the pump i
- t_i is the length of operation time of the pump

Pump	t_i	x_i
1	94.5	5
2	15.7	1
3	62.9	5
4	126	14
5	5.24	3
6	31.4	19
7	1.05	1
8	1.05	1
9	2.1	4
10	10.5	22

Choice of the prior

A conjugate gamma prior distribution is adopted for the failure rates:

$$\theta_i \sim Gamma(\alpha, \beta), \quad i = 1, ..., 10$$

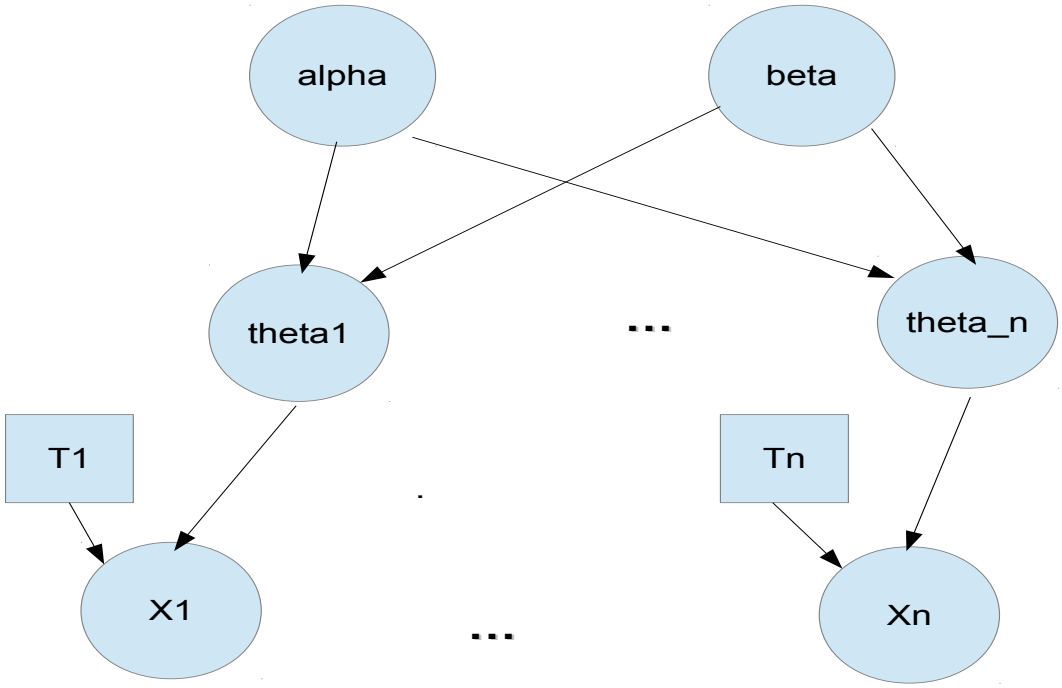
George *et al* (1993) assume the following prior specification for the hyper parameter a

$$\alpha \sim Exponential(1.0)$$
$$\beta \sim Gamma(0.1, 1)$$

Remarque

We can implement a Gibbs sampler to approximate the posterior distribution.

DAG



Définition du modèle

```
model
{
for (i in 1 : N) {
theta[i] ~ dgamma(alpha, beta)
lambda[i] <- theta[i] * t[i]
x[i] ~ dpois(lambda[i])
}
alpha ~ dexp(1)
beta ~ dgamma(0.1, 1.0)
}
```

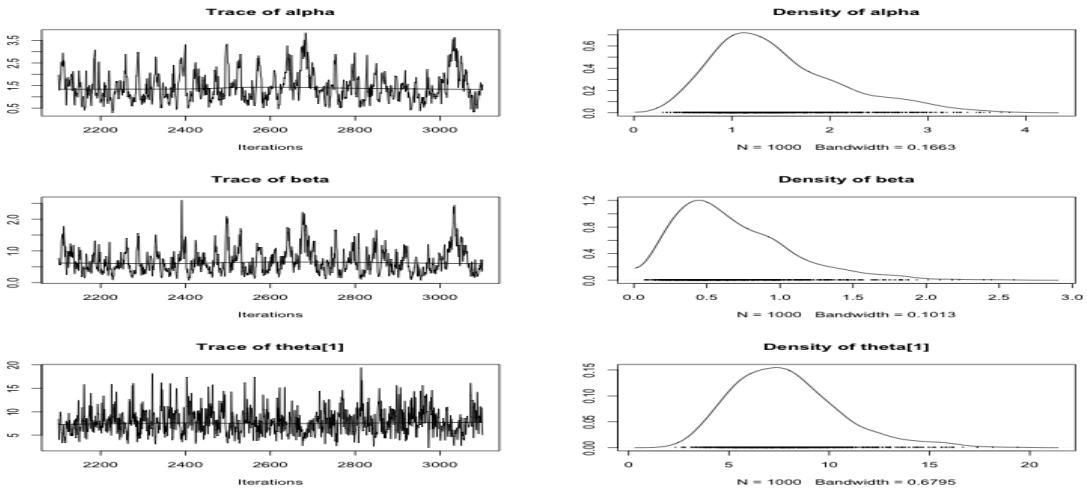
Ce code est stocké dans le fichier `model.bugs`

Compilation du modèle

```
N <- 10
t = c(94.3, 15.7, 62.9, 126, 5.24, 31.4, 1.05, 1.05, 2.1, 10.5)
x = c(5,  1,  5,   14,  3,19,  1,   1,4,22)

> jags <- jags.model('model.bugs',
+                    data = list('x' = x,'t' =t , 'N' = N),
+                    n.chains = 1,
+                    n.adapt = 1000,
+                    )

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 45
Initializing model
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100%
 > update(jags, 1000)
  |**************************************************| 100%
> z = coda.samples(jags,c('theta','alpha','beta'),1000)
  |**************************************************| 100%
```

Exploitation des résultats pour les sorties

```
> plot(z)
```



Statistiques sur la loi a posteriori

```
> summary(z)

Iterations = 3001:4000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
```

	Mean	SD	Naive SE	Time-series SE
alpha	0.69843	0.26009	0.0082248	0.0195038
beta	0.92433	0.54403	0.0172037	0.0373580
theta[1]	0.06016	0.02409	0.0007616	0.0006697
theta[2]	0.09889	0.07864	0.0024870	0.0030839
theta[3]	0.09140	0.03783	0.0011962	0.0011961
theta[4]	0.11585	0.03081	0.0009744	0.0008655
theta[5]	0.60821	0.31897	0.0100866	0.0100259
theta[6]	0.61247	0.14025	0.0044350	0.0048897
theta[7]	0.89937	0.72849	0.0230369	0.0227078
theta[8]	0.89025	0.72162	0.0228197	0.0241796
theta[9]	1.59729	0.76526	0.0241997	0.0301006
theta[10]	1.99623	0.41496	0.0131221	0.0127887

Statistiques sur la loi a posteriori – suite

```
2. Quantiles for each variable:
```

	2.5%	25%	50%	75%	97.5%
alpha	0.288329	0.51846	0.65789	0.84936	1.2993
beta	0.206124	0.53636	0.81742	1.20752	2.1961
theta[1]	0.018469	0.04237	0.05791	0.07463	0.1117
theta[2]	0.007171	0.04034	0.07858	0.13663	0.3047
theta[3]	0.030760	0.06416	0.08618	0.11584	0.1767
theta[4]	0.064556	0.09444	0.11211	0.13334	0.1851
theta[5]	0.157434	0.36772	0.55187	0.79335	1.3592
theta[6]	0.364537	0.51456	0.60387	0.70302	0.9099
theta[7]	0.086953	0.36652	0.70169	1.23917	2.7364
theta[8]	0.074552	0.38537	0.73600	1.18122	2.7251
theta[9]	0.467774	1.03672	1.44874	2.02527	3.5051
theta[10]	1.238676	1.70397	1.99607	2.27330	2.8236

Approximation des régions HPD

```
> HPDinterval(z)
[[1]]
                lower     upper
alpha      0.238151955 1.2314348
beta       0.127041180 1.9591916
theta[1]   0.014811045 0.1048188
theta[2]   0.001897323 0.2547897
theta[3]   0.024882522 0.1643409
theta[4]   0.055314737 0.1720302
theta[5]   0.105966924 1.2443890
theta[6]   0.353274820 0.8837728
theta[7]   0.003061533 2.3531108
theta[8]   0.010275707 2.2814622
theta[9]   0.369339389 3.1483068
theta[10] 1.167726211 2.7426627
attr(,"Probability")
[1] 0.95
```

Critère de Gelman et Rubin

Il faut simuler plusieurs chaines indépendantes (ici 4)

```
> jags <- jags.model('model.bugs',
+                    data = list('x' = x,'t' =t , 'N' = N),
+                    n.chains = 4,
+                    n.adapt = 1000,
+                    )
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 45
Initializing model

  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100%
> update(jags, 1000)
  |**************************************************| 100%
> z = coda.samples(jags,c('theta','alpha','beta'),1000)
  |**************************************************| 100%
```

Critère de Gelman et Rubin suite

```
> gelman.diag(z)
Potential scale reduction factors:

          Point est. Upper C.I.
alpha           1.00       1.01
beta            1.01       1.02
theta[1]        1.00       1.00
theta[2]        1.00       1.00
theta[3]        1.00       1.00
theta[4]        1.00       1.00
theta[5]        1.00       1.00
theta[6]        1.00       1.00
theta[7]        1.00       1.00
theta[8]        1.00       1.00
theta[9]        1.00       1.01
theta[10]       1.00       1.00

Multivariate psrf

1
```