# Calcul de postériore par programmation R d'un algorithme MCMC (Gibbs sampling et HM dans le Gibbs) et comparaison avec WinBugs : application au modèle campylobacter-poulet

**Isabelle Albert**
Met@risk, INRA
albert@inapg.inra.fr

**Judith Rousseau**
CEREMADE-Université Paris Dauphine
rousseau@ceremade.dauphine.fr

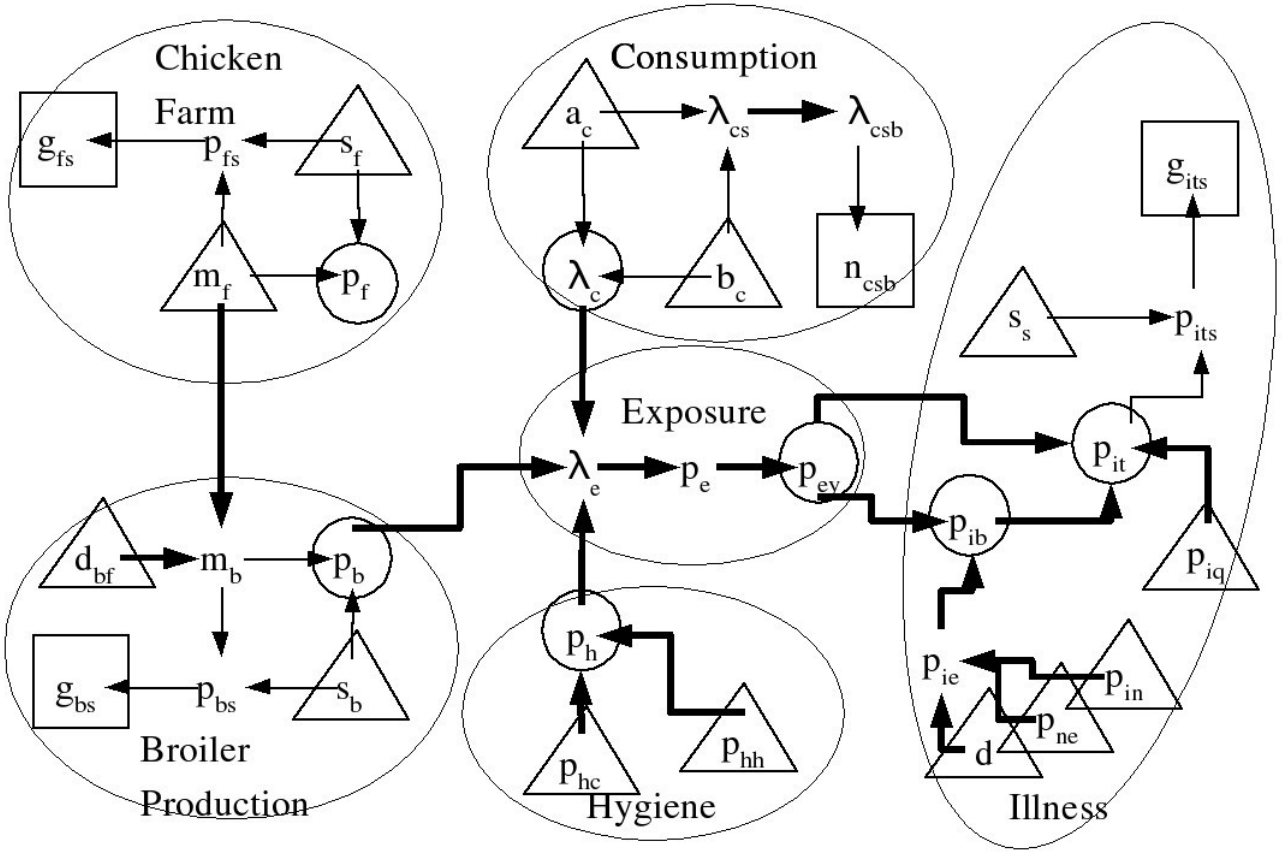
**Jean-Baptiste Denis**
MIA-Jouy-en-josas, INRA
jean-baptiste.denis@jouy.inra.fr

http://metarisk.inapg.inra.fr/UserFiles/File/Working%20paper%20serie/WP07-11.pdf

# Plan

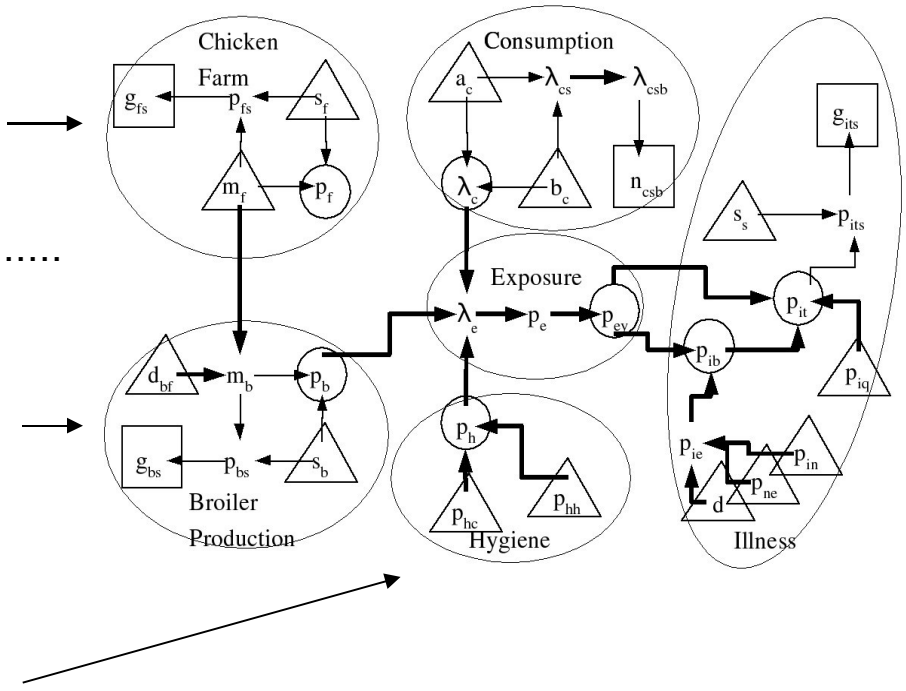
2. Définition initiale du modèle
   1.1 Structure générale du modèle (code WinBUGS)
   1.2 Les constantes
   1.3 Les distributions conditionnelles
3. Modèle utilisé pour l'algorithme
   2.1 Nouvelle structure
   2.2 Distributions locales nouvelles
4. Les densités conditionnelles complètes
5. Description de l'algorithme
6. Comparaisons des résultats
7. Conclusions

# 1. Définition initiale du modèle



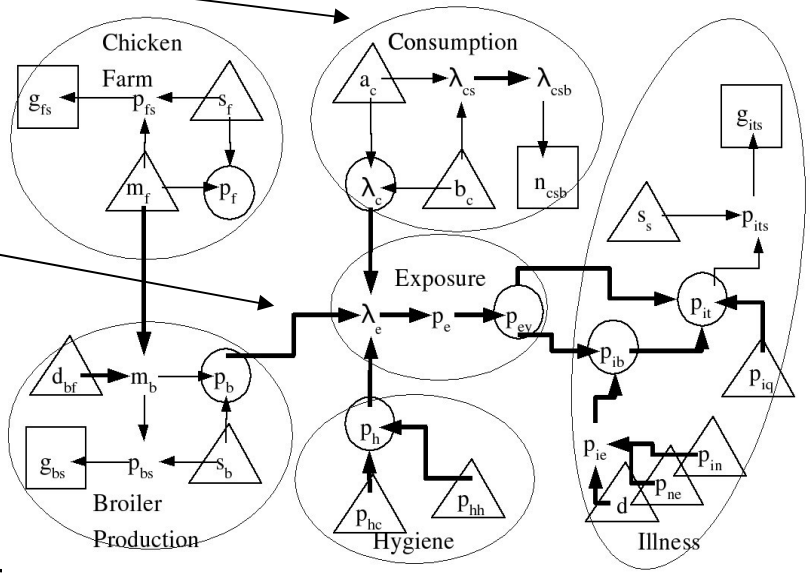
# 1.1 Le code WinBUGS

```
model   { # starting the model .....
### CORE MODEL
#########################################
### FLOCK MODULE .....
p.f <- exp(lp.f)/(١+exp(lp.f));                  #vi
lp.f ~ dnorm(m.f, tau.f);                        #..
m.f ~ dnorm(٠.٠,٢٠.٦٦);                          #pa
s.f ~ dunif(٠,٠.٢); tau.f <- ١/(s.f*s.f);        #pa
### TRANSFORMATION MODULE .....
p.b <- exp(lp.b)/(١+exp(lp.b));                  #vi
lp.b ~ dnorm(m.b, tau.b);                        #..
m.b <- m.f + d.bf;                               #cv
d.bf ~ dnorm(٠.١,٢٠٦.٦)I(٠,);                    #pa
s.b ~ dunif(٠,٠.٢); tau.b <- ١/(s.b*s.b);        #pa
### HYGIENE MODULE .....
p.h <- p.hc * p.hh;                              #vi
p.hc ~ dbeta(٨,٨);                               #pa
p.hh ~ dbeta(٨,٢٨);                              #pa
```



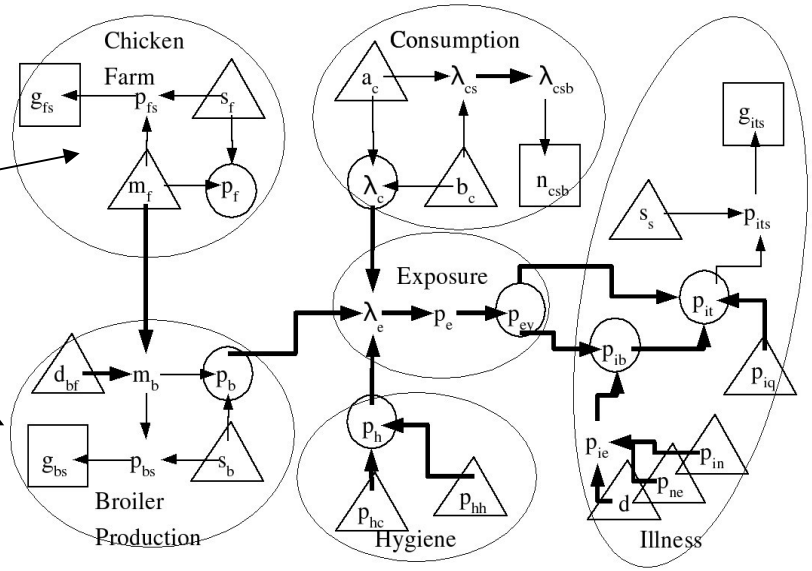
# 1.1 Le code WinBUGS

```
### CONSUMPTION MODULE .....................................
lambda.c ~ dgamma(ab.c,b.c);                          #vi
ab.c <- a.c*b.c;                                      #..
a.c ~ dgamma(٤,٤);                                    #pa
b.c ~ dgamma(١٠,١٠);                                  #pa
### EXPOSURE MODULE .....................................
p.ey <- ١ - pow((١-p.e),١٣);                          #vi
p.e <- ١ - exp(-lambda.e);                            #cv
lambda.e <- p.b * p.h * lambda.c;                     #cv
### ILLNESS MODULE .....................................
p.ib <- p.ey * p.ie;                                  #vi
p.it <- p.ib / (١ - (١-p.iq)*(١-p.ey));               #vi
p.ie <- (١-pow(١-p.ne,d)) * p.in;                     #cv
d <- vd[c.d];                                         #cv
p.ne ~ dbeta(٠.٠٢٤,٠.٠١١)I(٠.٠٠٠٠١,٠.٩٩٩٩٩٩);        #pa
p.in <- ٠.٣٣;                                         #pa
p.iq ~ dbeta(٩,٣٠);                                   #pa
c.d ~ dcat(c.di[]);                                   #..
```



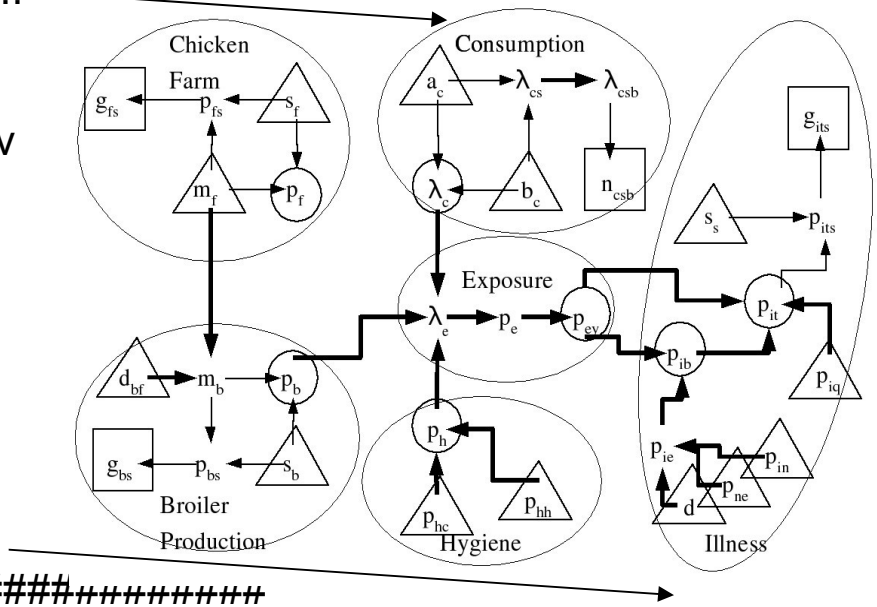
# 1.1 Le code WinBUGS

```
### AUGMENTED MODEL FOR DATA INCORPORATION
###############
### FLOCK MODULE .....................................
for (i.fs in 1:n.fst) {                              #..
  g.fs[i.fs] ~ dbin(p.fs[i.fs],n.fs[i.fs]);            #da
  p.fs[i.fs] <- exp(lp.fs[i.fs]) /                     #cv
                (1 + exp(lp.fs[i.fs]));                #..
  lp.fs[i.fs] ~ dnorm(m.f,tau.f);                      #..
}                                                    #..
### TRANSFORMATION MODULE .....................
for (i.bs in 1:n.bst) {                              #..
  g.bs[i.bs] ~ dbin(p.bs[i.bs],n.bs[i.bs]);            #da
  p.bs[i.bs] <- exp(lp.bs[i.bs]) /                     #cv
                (1 + exp(lp.bs[i.bs]));                #..
  lp.bs[i.bs] ~ dnorm(m.b,tau.b);                      #..
}                                                    #..
```



# 1.1 Le code WinBUGS

```
### HYGIENE MODULE .....................................
### CONSUMPTION MODULE ..................................
for (i.cs in 1:n.cst) {                          #..
  n.csb[i.cs] ~ dpois(lambda.csb[i.cs]);          #da
  lambda.csb[i.cs] <- n.cs[i.cs]*lambda.cs[i.cs];  #cv
  lambda.cs[i.cs] ~ dgamma(ab.c,b.c);             #cv
}                                                 #..
### EXPOSURE MODULE .....................................
### ILLNESS MODULE ......................................
g.its ~ dbin(p.its,n.its);                        #da
logit(p.its) <- logit(p.it) + err;                #cv
s.s <- 1; tau.s <- 1/(s.s*s.s);                   #pa
err ~ dnorm(0,tau.s);                             #..
} # ending the model ##############################
```



# 1.1 Le code WinBUGS – Les données

```
list( c.di=c(٠.٥٠٠,٠.١٦٣,٠.٢٢٢,٠.٠٩٧,٠.٠١٨),
vd=c(١,٢,١٠,١٠٠,٣٠٠))

list(
# nombre de lots de poulets examinés
n.fst=16,
# nombre de poulets par lot
n.fs =c(287, 79, 100, 8911, 75, 59, 125, 155, 403, 398, 112, 62, 187, 100, 22, 450),
# nombre de lots de carcasses examinés
n.bst=14,
# nombre de carcasses par lot
n.bs=c(35, 82, 2925, 410, 251, 691, 203, 79, 50, 97, 180, 708, 994, 858),
# nombre de ménages de l'enquête
n.cst=4770,
# nombre de periodes d'analyse du ménage
n.cs=c(7, 7, 7, 3, 4, 8, 8, 5, 5, 4,
```

# 1.1 Le code WinBUGS – Les données

```
# nombre de volontaires dans l'expérimentation
n.its=4026,
#
###
#
# chargement des données  ################################## ((2))
# nombres de poulets contaminés par lots
g.fs=c( 77, 29,  40, 3787, 32, 29,  63,  80, 228, 226,
        64, 50, 153,  90, 22, 294),
# nombres de carcasses contaminées par lots
g.bs=c(28, 18, 1082, 157,  99, 333, 163, 70, 43, 78, 146,
       291, 338,  80),
# nombres de poulets achetés par ménage
n.csb=c(1, 2, 12, 0, 1, 6, 3, 1, 3, 0, 5, 6, 28,


 # nombres de campylobactérioses équivalent années
 g.its=٣٢
 )
```

# 1.2 Les constantes (qui spécifient les distributions sur les nœuds ancêtres)

$$
\begin{aligned}
(k_{mf1}, k_{mf2}, k_{sf1}, k_{sf2}) &= (0, \sqrt{1/20.66}, 0, 0.2) \\
(k_{dbf1}, k_{dbf2}, k_{sb1}, k_{sb2}) &= (0.1, \sqrt{1/206.6}, 0, 0.2) \\
(k_{ac1}, k_{ac2}, k_{bc1}, k_{bc2}) &= (4, 4, 10, 10) \\
(k_{hc1}, k_{hc2}, k_{hh1}, k_{hh2}) &= (8, 8, 8, 28) \\
(k_{pne1}, k_{pne2}, k_{pin}, k_{piq1}, k_{piq2}) &= (0.024, 0.011, 0.33, 9, 30) \\
(k_{d1}, k_{d2}, k_{d3}, k_{d4}, k_{d5}) &= (0.500, 0.163, 0.222, 0.097, 0.018) \\
(d_1, d_2, d_3, d_4, d_5) &= (1, 2, 10, 100, 300) \\
k_{ss} &= 1
\end{aligned}
$$

# 1.3 Les distributions conditionnelles

$$
\begin{aligned}
m_f &\sim N(k_{mf1}, k_{mf2}) \\
s_f &\sim U(k_{sf1}, k_{sf2}) \\
logit\,(p_f) \mid m_f, s_f &\sim N(m_f, s_f) \\
d_{bf} &\sim N(k_{dbf1}, k_{dbf2})\mathbf{1}_{[0,+\infty[} \\
s_b &\sim U(k_{sb1}, k_{sb2}) \\
m_b \mid m_f, m_{db} &= m_f + d_{bf} \\
logit\,(p_b) \mid m_b, s_b &\sim N(m_b, s_b) \\
logit\,(p_{fs}) \mid m_f, s_f &\sim N(m_f, s_f) \\
logit\,(p_{bs}) \mid m_b, s_b &\sim N(m_b, s_b) \\
g_{fs} \mid p_{fs} &\sim Binom(n_{fs}, p_{fs}) \\
g_{bs} \mid p_{bs} &\sim Binom(n_{bs}, p_{bs})
\end{aligned}
$$

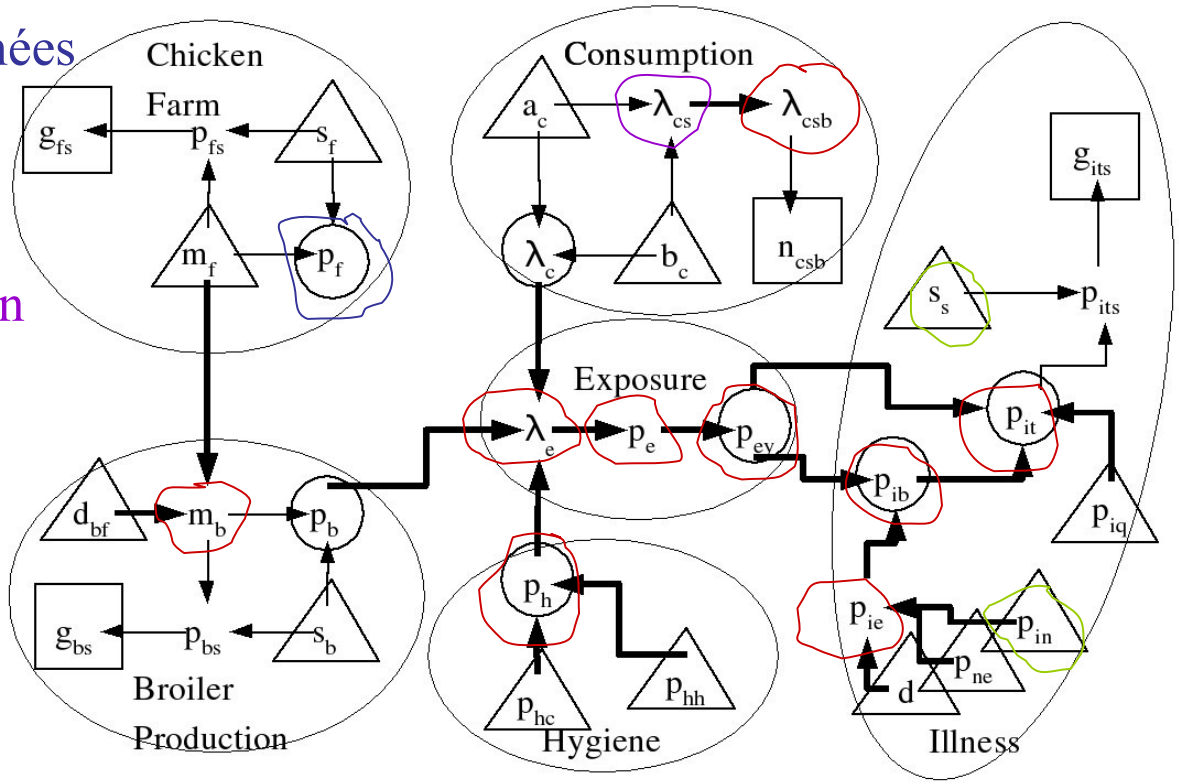
$$
\begin{aligned}
a_c &\sim Gamma\,(k_{ac1}, k_{ac2}) \\
b_c &\sim Gamma\,(k_{bc1}, k_{bc2}) \\
\lambda_c \mid a_c, b_c &\sim Gamma\,(a_c b_c, b_c) \\
\lambda_{cs} \mid a_c, b_c &\sim Gamma\,(a_c b_c, b_c) \\
n_{csb} \mid \lambda_{cs}, n_{cs} &\sim Poisson\,(n_{cs}\lambda_{cs}) \\
p_{hc} &\sim Beta\,(k_{hc1}, k_{hc2}) \\
p_{hh} &\sim Beta\,(k_{hh1}, k_{hh2}) \\
p_h \mid p_{hc}, p_{hh} &= p_{hc} p_{hh} \\
\lambda_e \mid p_b, p_h, \lambda_c &= p_b p_h \lambda_c
\end{aligned}
$$

# 1.3 Les distributions conditionnelles

$$
\begin{aligned}
p_e \mid \lambda_e &= 1 - \exp(-\lambda_e) \\
p_{ey} \mid p_e &= 1 - (1 - p_e)^{13} \\
p_{ne} &\sim Beta(k_{pne1}, k_{pne2}) \\
p_{in} &= k_{pin} \\
p_{iq} &\sim Beta(k_{piq1}, k_{piq2}) \\
d &\sim Empirical(k_d) \\
p_{ie} \mid p_{in}, p_{ne}, d &= \left(1 - (1 - p_{ne})^d\right) p_{in} \\
p_{ib} \mid p_{ey}, p_{ie} &= p_{ey} p_{ie} \\
p_{it} \mid p_{ey}, p_{ib}, p_{iq} &= p_{ib} \left(1 - (1 - p_{iq})(1 - p_{ey})\right)^{-1} \\
s_s &= k_{ss} \\
logit(p_{its}) \mid p_{it}, s_s &\sim N(logit(p_{it}), s_s) \\
g_{its} \mid p_{its} &\sim Binom(n_{its}, p_{its})
\end{aligned}
$$

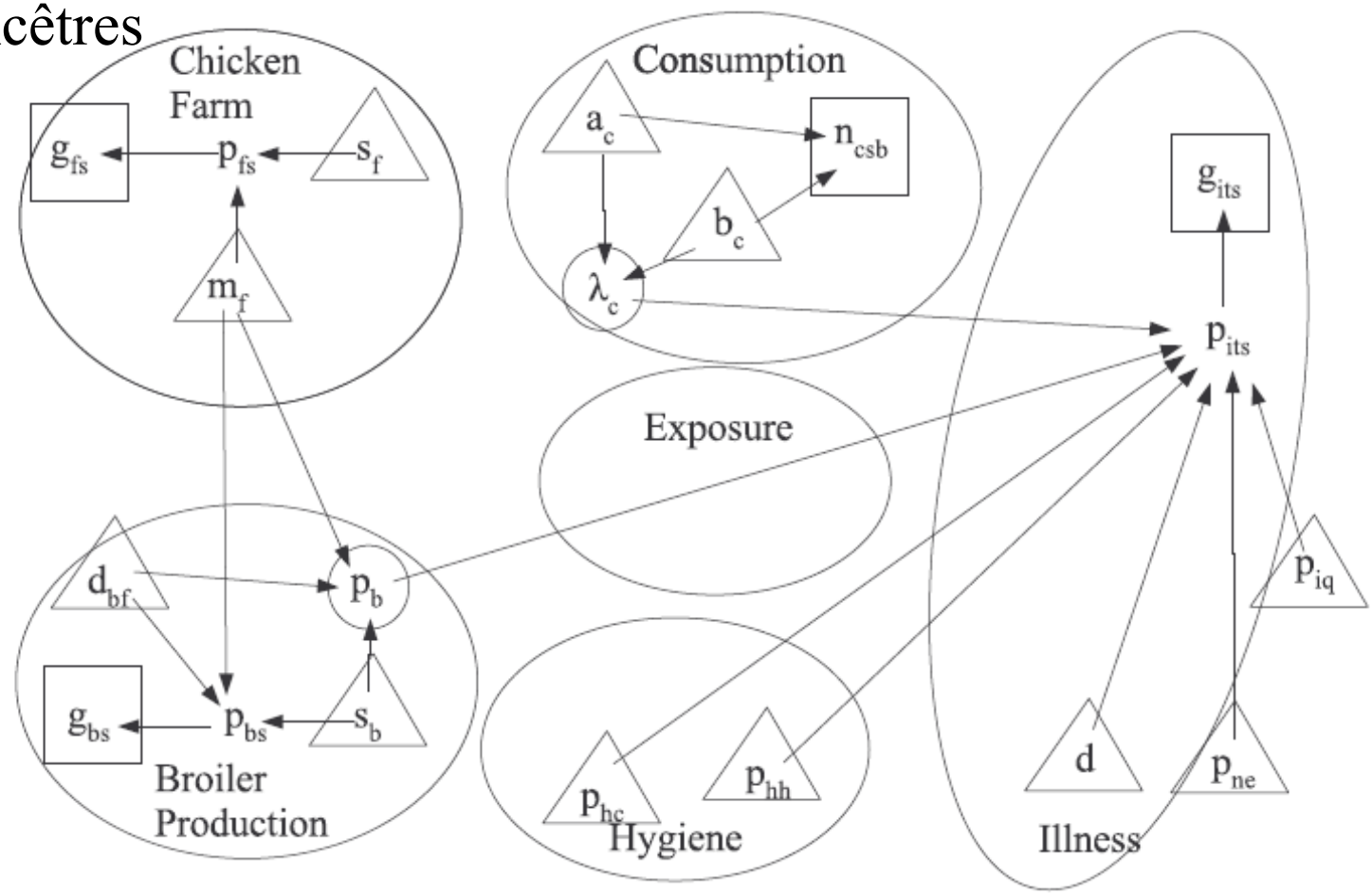
# 2. Modèle utilisé pour la programmation R

- Suppression des relations fonctionnelles (relais notationnels ou variables d'intérêt) et les nœuds ancêtres constants
- Suppression des variables d'intérêt non utiles à la définition de la vraisemblances des données
- Supprimer (ou ajouter) des variables intermédiaires (diminution ou augmentation de la dimension aléatoire du modèle)



# 2.1 Modèle utilisé pour la programmation R

- Conservation de la modélisation des données et des paramètres ancêtres



## 2.2 Les distributions conditionnelles nouvelles

$$
\begin{aligned}
m_f &\sim N(k_{mf1}, k_{mf2}) \\
s_f &\sim U(k_{sf1}, k_{sf2}) \\
d_{bf} &\sim N(k_{dbf1}, k_{dbf2})\mathbf{1}_{[0,+\infty[} \\
s_b &\sim U(k_{sb1}, k_{sb2}) \\
logit\,(p_b) \mid m_f, d_{bf}, s_b &\sim N(m_f + d_{bf}, s_b) \\
logit\,(p_{fs}) \mid m_f, s_f &\sim N(m_f, s_f) \\
logit\,(p_{bs}) \mid m_f, d_{bf}, s_b &\sim N(m_f + d_{bf}, s_b) \\
g_{fs} \mid p_{fs} &\sim Binom(n_{fs}, p_{fs}) \\
g_{bs} \mid p_{bs} &\sim Binom(n_{bs}, p_{bs}) \\
a_c &\sim Gamma\,(k_{ac1}, k_{ac2})
\end{aligned}
$$

**Densité jointe = produit de ces distributions**

$$
\begin{aligned}
b_c &\sim Gamma\left(k_{bc1}, k_{bc2}\right) \\
\lambda_c \mid a_c, b_c &\sim Gamma\left(a_c b_c, b_c\right) \\
n_{csb} \mid a_c, b_c &\sim BinoNeg\left(a_c b_c, \frac{b_c}{n_{cs} + b_c}\right) \\
p_{hc} &\sim Beta\left(k_{hc1}, k_{hc2}\right) \\
p_{hh} &\sim Beta\left(k_{hh1}, k_{hh2}\right) \\
p_{ne} &\sim Beta\left(k_{pne1}, k_{pne2}\right) \\
p_{iq} &\sim Beta\left(k_{piq1}, k_{piq2}\right) \\
d &\sim Empirical\left(k_d\right)
\end{aligned}
$$

**Distributions conditionnelles complètes d'un paramètre = produit des distributions où il apparaît**

$$
\begin{aligned}
logit\left(p_{its}\right) \mid p_b, p_{hc}, p_{hh}, \lambda_c, p_{ne}, d &\sim N\left(logit\left(\frac{\left[1 - \exp\left(-13 p_b p_{hc} p_{hh} \lambda_c\right)\right]\left[1 - \left(1 - p_{ne}\right)^d\right] k_{pin}}{1 - \left(1 - p_{iq}\right)\exp\left(-13 p_b p_{hc} p_{hh} \lambda_c\right)}\right), k_{ss}\right) \\
g_{its} \mid p_{its} &\sim Binom\left(n_{its}, p_{its}\right)
\end{aligned}
$$

# 3. *(m_f)*

$$\lfloor m_f \mid . \rfloor = -\frac{1}{2}\frac{(m_f - k_{mf1})^2}{k_{mf2}^2} - \frac{1}{2}\left(\sum_s \frac{(\ln \frac{p_{fs}}{1-p_{fs}} - m_f)^2}{s_f^2} + \frac{(logit\,(p_b) - m_f - d_{bf})^2 + \sum_s (logit\,(p_{bs}) - m_f - d_{bf})^2}{s_b^2}\right)$$

Forme quadratique en *m_f,* signature de la loi normale, de moyenne :

$$\mu_{mf|.} = \frac{k_{mf1}s_f^2 s_b^2 + k_{mf2}^2 s_b^2 \sum_s \left(\ln \frac{p_{fs}}{1-p_{fs}}\right) + k_{mf2}^2 s_f^2 \left[\sum_s \left(\ln \frac{p_{bs}}{1-p_{bs}} - d_{bf}\right) + logit\,(p_b) - d_{bf}\right]}{k_{mf2}^2(s_b^2 n_f + s_f^2(n_b+1)) + s_f^2 s_b^2}$$

et de variance :

$$\sigma_{mf|.}^2 = \frac{s_b^2 s_f^2 k_{mf2}^2}{k_{mf2}^2(s_b^2 n_f + s_f^2(n_b+1)) + s_f^2 s_b^2}$$

***(s_f)***

$$\lfloor s_f \mid . \rfloor = \mathbf{1}_{[k_{sf1}, k_{sf2}]}(s_f) \left( -n_f \lg s_f - \frac{\sum_s (\ln \frac{p_{fs}}{1-p_{fs}} - m_f)^2}{2 s_f^2} \right)$$

Distribution qui ne semble pas de forme connue et qui dépend de $m_f, (p_{fs})$.

***(d_bf)*** loi normale restreinte au support positif

***(s_b), (p_b), (p_fs), (p_bs), (a_c), (b_c), (λ_c), (p_hc), (p_hh), (p_ne), (p_iq), (p_its)*** pas de forme connue

***(d)***

$$P(D = d_q) \propto k_{dq} \exp \left( - \frac{\left\{ logit(p_{its}) - logit\left( \frac{[1-\exp(-13 p_b p_{hc} p_{hh} \lambda_c)]\left[1-(1-p_{ne})^{d_q}\right] k_{pin}}{1-(1-p_{iq})\exp(-13 p_b p_{hc} p_{hh} \lambda_c)} \right) \right\}^2}{2 k_{ss}^2} \right)$$

# 4. Description de l'algorithme (Gibbs sampling)

- Tirages successifs dans les lois conditionnelles complètes
- Initialisation (*t=0*): tirage dans les distributions conditionnelles
- Puis passage de l'itération *t* à *t+1* :
  - 1. mise à jour de $m_f$. Tirage direct dans une loi normale $N(\mu_{m_{f|.}}, \sigma^2_{m_{f|.}})$ où les valeurs utilisées pour les autres paramètres sont $s_{f,(t-1)}$, $d_{bf,(t-1)}$, $s_{b,(t-1)}$, $(p_{fs,(t-1)})$
  - 2. mise à jour de *s_f* par Hastings-Metropolis

    2.1 Tirage d'un candidat *s_f** d'une distribution normale centrée sur *s_f, (t-1)* et dont la variance est un paramètre de pilotage à fixer et tronquée sur le support de la priore (distribution instrumentale)

# 4. Description de l'algorithme

## 2. mise à jour de *s_f* par Hastings-Metropolis (suite)

2.2 $s\_f^*$ est accepté avec la proba :

$$\rho_a = \min\left(1, \frac{[X_k^* \mid ...].\pi\left(X_k^{(t-1)} \mid X_k^*\right)}{\left[X_k^{(t-1)} \mid ...\right].\pi\left(X_k^* \mid X_k^{(t-1)}\right)}\right)$$

i.e. Si *runif(1)* < *exp* ( $\lfloor s_f^* \rfloor$ - $\lfloor s_{f,(t-1)} \rfloor$ ) alors *s_f(t)= s_f** sinon *s_f(t)=s_f(t-1)*

$$\lfloor s_f \mid . \rfloor = \mathbf{1}_{[k_{sf1}, k_{sf2}]}(s_f)\left(-n_f \lg s_f - \frac{\sum_s (\ln \frac{p_{fs}}{1-p_{fs}} - m_f)^2}{2s_f^2}\right)$$

- et ainsi de suite,
  - Attention aux mises à jour des probabilités par d'un tirage normal sur leur logit
    - *Si* $logit(p^*) \sim N(logit(p^{t-1}), s\_$

$$\rho_a = \min\left(1, \frac{\pi(p^*).p^*(1-p^*)}{\pi(p^{t-1}).p^{t-1}(1-p^{t-1})}\right)$$

*Alors dans le HM:*

# 4. Description de l'algorithme

- mise à jour de *d* : tirage d'une multinomiale des nouvelles valeurs *d_t* avec les proba conditionnelles complètes associées à chaque valeur de *d*

## Sélection des valeurs simulées

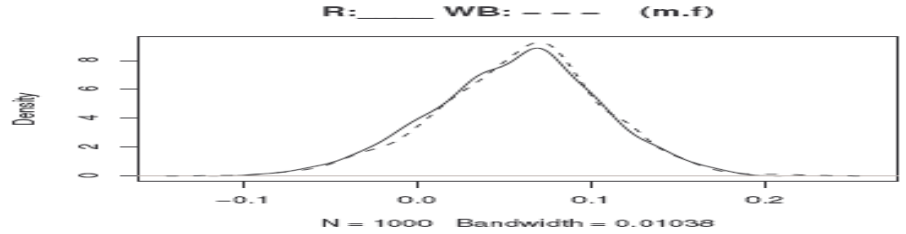
- T itérations réalisées ; Ne sont retenues que celles après burn-in (temps de convergence) et éclaircissement (thin) (pour limiter les dépendances entre les itérations successives)

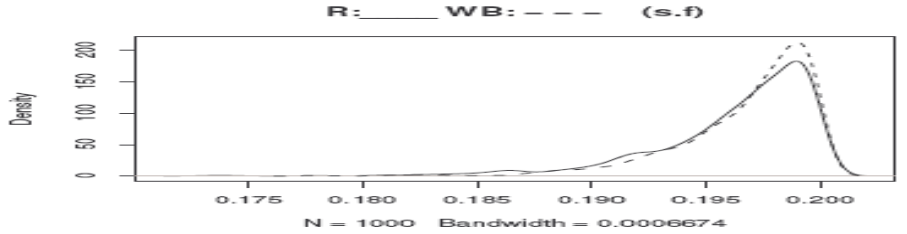
## Simulation des variables dérivées

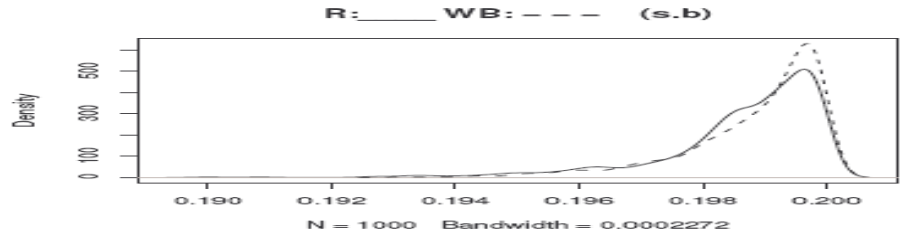
- Ici *p_f, p_ey, p_ib, p_it*

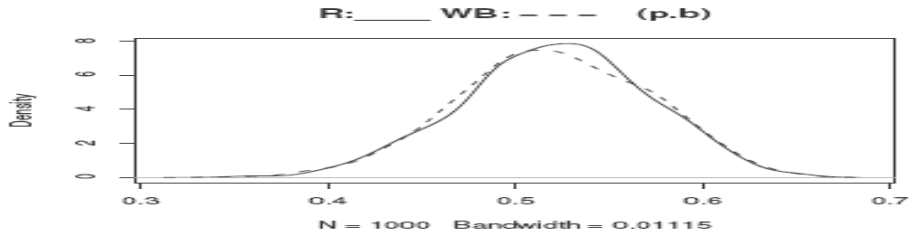
# 5. Comparaison des résultats avec Winbugs

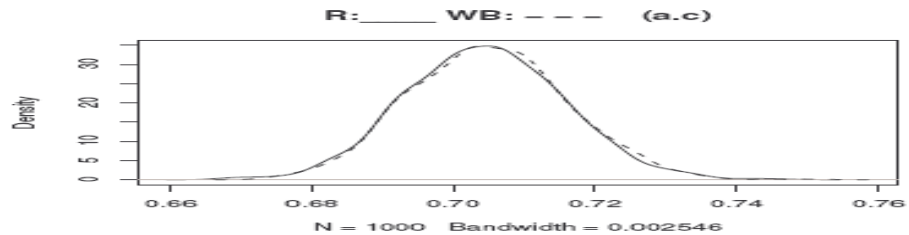
(burn-in: $10^4$, thin=100, 1000 gardés, total=110000)

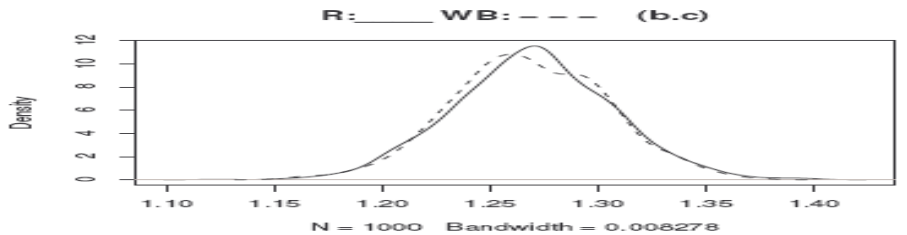


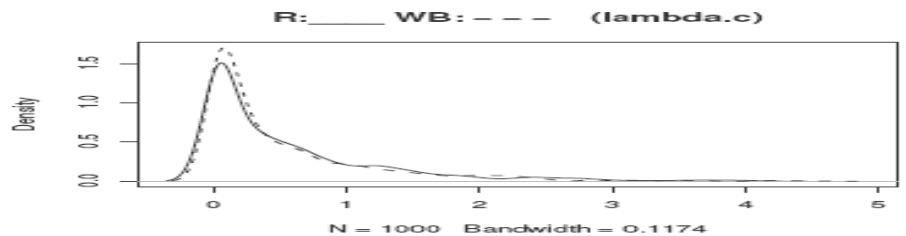



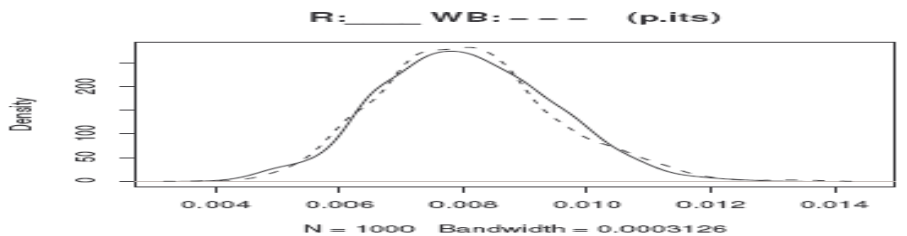



# Taux d'acceptation des algorithmes HM (« théorie » entre 20% et 40%)

| s.f | p.f1 | p.f2 | p.f3 | p.f4 | p.f5 | p.f6 | p.f7 |
|---|---|---|---|---|---|---|---|
| 0.37 | 0.61 | 0.70 | 0.69 | 0.18 | 0.70 | 0.72 | 0.67 |
| p.f8 | p.f9 | p.f10 | p.f11 | p.f12 | p.f13 | p.f14 | p.f15 |
| 0.65 | 0.55 | 0.56 | 0.68 | 0.71 | 0.66 | 0.70 | 0.75 |
| p.f16 | s.b | p.b | p.b1 | p.b2 | p.b3 | p.b4 | p.b5 |
| 0.55 | 0.29 | 0.38 | 0.60 | 0.56 | 0.18 | 0.40 | 0.45 |
| p.b6 | p.b7 | p.b8 | p.b9 | p.b10 | p.b11 | p.b12 | p.b13 |
| 0.33 | 0.49 | 0.57 | 0.59 | 0.55 | 0.50 | 0.33 | 0.30 |
| p.b14 | a.c | b.c | lambda.c | p.hc | p.hh | p.ne | p.iq |
| 0.39 | 0.12 | 0.13 | 0.47 | 0.38 | 0.38 | 0.59 | 0.38 |
| p.its | | | | | | | |
| 0.39 | | | | | | | |

# Rapidité des calculs

- Équivalente sur notre exemple (environ 1 heure)
  - Mais problème de stabilité de la Bêta(0.024,0.011) avec WinBUGS

# 6. Conclusion

## Programmation R

- De nombreuses sources d'erreurs possibles

- Nécessite une programmation "minutieuse"

- Peut-être plus longue si que des HM (pas d'algorithme de rejet adaptatif ou de slice sampler)

## Mais

- Meilleur contrôle des paramètres de l'algorithme

- Meilleure stabilité numérique

- Pas les limites distributionnelles et fonctionnelles de WinBUGS