JAGS: Just Another Gibbs Sampler

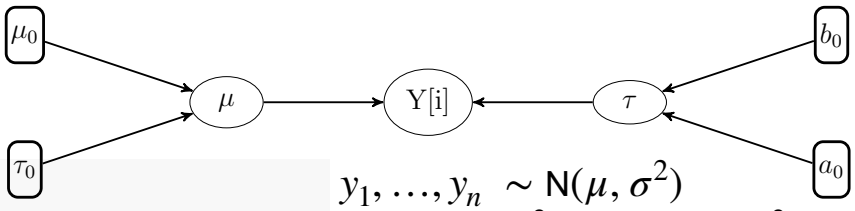
`glm.out <- glm(`$y \sim x$`, family=., data=.)` $y \sim x$ is the description of a model, queried via extractor functions, `summary(), coef, vcov`

```
library(rjags)
m <- jags.model("code",data,inits,n.chain=2)
list.samplers(m)
# Burn-in
updates(m,n.iter=4000)
```

m is not a fitted model, it is a dynamic object that can be queried to generate from the posterior; **"code"** name of the file containing a description of the model in BUGS language; **data** named list of data for observed variables; **inits** is a list of lists of initial values, one for each chain; **n.chain** number of chains to run; **updates** is used for adaptation, burn-in

To generate samples from a given model **m** `x <- coda.samples(m, variable.names=".", n.iter=1000)`
• `x` is an object of class mcmc.list -> `coda` • `coda.samples` is a wrapper function for `"jags.samples"`

NORMAL EXAMPLE IN JAGS



$y_1, \ldots, y_n \sim \mathrm{N}(\mu, \sigma^2)$
$\mu \sim \mathrm{N}(\mu_0, \sigma_0^2), \mu_0 = -3, \sigma_0^2 = 4$
precision:
$\tau = 1/\sigma^2$ (1/variance) $\sim \mathrm{G}(a_0, b_0)$
$a_0 = 1.6, b_0 = 0.4$
R/Stan: N(mean, sd)
BUGS/JAGS/INLA:
$\mathrm{N}(\mu, \tau^{-1}), \tau^{-1} = \sigma^2, \tau = \frac{1}{\sigma^2}$

INLA used to compare

```
# Define the parameters of the prior distributions
mu0 <- -3
sigma2_0 <- 4
a0 <- 1.6
b0 <- 0.4
suppressPackageStartupMessages(library(INLA))
formula <- y ~ 1
inla.output <- inla(formula,data=data.frame(y=y),
                    control.family = list(hyper =
                                            list(prec = list(prior="loggamma",
                                                             param=c(a0,b0)))),
                    control.fixed = list(mean.intercept=mu0,
                                         prec.intercept=1/sigma2_0))
```

```
set.seed(44566)
suppressPackageStartupMessages(library(rjags))
suppressPackageStartupMessages(library(coda))
wb_data <- list( N=30,
                 y=c(3.048,2.980,2.029,7.249,-0.259,3.061,4.059,6.370,7.902,1.926,
                     9.094,10.489,-0.384,-3.096,2.315,5.830,-1.542,-1.544,5.714,
                     -5.182,3.828,-4.038,2.169,5.087,-0.201,4.880,3.302,3.859,
                     11.144,5.564)
)
wb_inits <- list( mu=-0.2381084, inv_sigma2=0.3993192 )
modelString = " # open quote for modelString
model{
# likelihood
for (i in 1:N){
y[i] ~ dnorm( mu, inv_sigma2 )
}
# Priors
mu ~ dnorm( -3, 0.25 ) # prior for mu N(mu0, prec=1/sigma2_0)
inv_sigma2 ~ dgamma( 1.6, 0.4 ) # prior for precision G(a0, b0)
# transformations
# deterministic definition of variance
sigma2 <- 1/inv_sigma2

# deterministic definition of standard deviation
sigma <- sqrt(sigma2)
}
" # close quote for modelString
writeLines(modelString, con="TempModelexe3.txt") # write to a file
```

JAGS CODE

```
model {
# likelihood
for(i in 1:N) {
Y[i] ~ dnorm(mu, tau)}
# priors
mu ~ dnorm(-3, 0.25) # (mu_0, tau_0)
tau ~ dgamma(1.6, 0.4) # (a_0, b_0)
}
```

list.factories and **set.factories** show and control over the status of factories in JAGS modules. **jags.modules** shows the names of the currently loaded modules and also loads or unloads JAGS modules.

```
list.factories(type = "rng")
##          factory status
## 1 base::BaseRNG   TRUE
```

```
load.module("glm")
list.modules()
## [1] "basemod"
    "bugs"    "glm"
unload.module("glm")
```

```
list.factories(type = "monitor")
##             factory status
## 1 base::Variance   TRUE
## 2     base::Mean   TRUE
## 3    base::Trace   TRUE
```

```
list.factories(type = "sampler")
##              factory status
## 1  bugs::BinomSlice   TRUE
## 2         bugs::RW1   TRUE
## 3    bugs::Censored   TRUE
```

```
set.factory(name = "base::Slice",
            type = "sampler", state = FALSE)
```

```
# model initiation
set.seed(44566)
model.jags <- jags.model(
  file = "TempModelexe3.txt",
  data = wb_data,
  inits = wb_inits,
  n.chains = 1,
  n.adapt = 4000
)
update(model.jags, n.iter = 4000) # burn-in
# sampling
fit.jags.coda <- coda.samples(
  model = model.jags,
  variable.names = c("mu", "sigma2", "inv_sigma2"),
  n.iter = 10000,
  thin = 1
)
summary(fit.jags.coda)
plot(fit.jags.coda)
```

one chain

```
wb_inits <- function() {
  list(mu = rnorm(1),
       inv_sigma2 = runif(1)
  )
}
# model initialisation
model.jags <- jags.model(
  file = "TempModelexe3.txt",
  data = wb_data,
  inits = wb_inits,
  n.chains = 4,
  n.adapt = 4000
)
```

several chains

```
effectiveSize(fit.jags.coda)
lapply(fit.jags.coda, effectiveSize)
gelman.diag(fit.jags.coda,autoburnin=TRUE)
gelman.plot(fit.jags.coda,autoburnin=TRUE)
geweke.diag(fit.jags.coda) heidel.diag(fit.jags.coda)
geweke.plot(fit.jags.coda)raftery.diag(fit.jags.coda)
coda:::traceplot(fit.jags.coda)
```

CODA

```
# "DIC" penalised expected deviance computation
dic1<-dic.samples(model=model.jags, n.iter=1000, type="popt")
```

Steps (JAGS follows it): • initialization • burn-in 1,…,b • monitoring (sampling) b+1, . . . , b+M (assumption stationary)

Q1 What should I use for starting values?
1. two (or four) different starting values 2. Idea: Start/initialize the chain somewhere near a measure of center of the relevant posterior distribution (mean, mode) (use maximum likelihood justified by weakly informative priors). 3. Remark: This can be problematic if the posterior is multimodal. For example, in a bimodal normal mixture $Y = \alpha X_1 + (1 - \alpha)X_2$.

Q2 **Frequentistic Hypothesis testing:** How long should burn-in period be? (How do you know when the Markov chain reaches equilibrium (stationary distribution)?) **WE DON'T KNOW**
• We can only look for evidence that it has not converged • This is the usual situation in hypothesis testing • But the consequences of the type II error are severe: Invalid INFERENCE

Gelman / Rubin / Brooks BGR (Convergence to ergodic average)

$$\sqrt{\widehat{R}} \approx \sqrt{\frac{W+B}{W}} \to 1, \quad \text{if} \quad B \to 0.$$

Idea: Run multiple chains from widely dispersed starting values and preform an Analysis of Variance to see if the between-chain variability (B) is large in relation to the average variability within (W + B) the (pooled) chain (if so this would indicate more than one mode). The idea is that if separate chains have not mixed well, the variance of all chains (W + B) taken together (pooled) should be higher than the variance of individual chains (W). Run many (at least 2 short chains and compare late parts (second half) of the chains).
Chains mix well and converge, if $\hat{R} \to 1$. $\hat{R}$ is also called psrf the "potential scale reduction factor". • If chains have not converged, they will be over-dispersed • No testing • "Shrink factor" $\hat{R}$ and Upper Confidence Interval • Uses normal approximation to derive $\hat{R}$.
Vehtari et al. [2021]: traditional $\hat{R}$ can fail to correctly diagnose convergence failures when the chain has a heavy tail or when the variance varies across the chains and proposed an alternative rank-based diagnostic. They recommend that at least four chains should be run by default and the threshold applied to $\hat{R}$ should be 1.01($\hat{R}$ < 1.01). They only recommend relying on the $\hat{R}$ estimate to make decisions about the quality of the chains if the ESS is large enough.

$$\widehat{R} \approx \sqrt{1 + \frac{\texttt{n.chain}}{\text{ESS}}}$$

Vats and Knudson [2021]: a cutoff of $\hat{R} \leq 1.1$ is too high to yield reasonable estimates of target quantities. They show that $\hat{R}$ = 1.1 corresponds to ESS = 5 × n.chain (5 independent samples per chain), which is too low to estimate the mean with any reasonable certainty. In contrast, $\hat{R}$ = 1.01 corresponds to ESS = 50 × n.chain (50 independent samples per chain), which is more appropriate for estimation of the mean.

Geweke (Convergence to stationarity) Detects cases when equilibrium has not been reached.

$$Z = \frac{\bar{\theta}^{*A} - \bar{\theta}^{*B}}{\sqrt{\text{Var}(\theta^{*A}) + \text{Var}(\theta^{*B})}}$$

Idea: • consider a single long run • test for equality of means between early and late sections of the chain • essentially, two independent sample t-tests. Example : Assume: Ⓐ corresponds to early 10 % (frac1) of the chain Ⓑ corresponds to next 50 % (frac2) of the chain. Does | Z |> 2? **geweke.plot**: 2.5 % of Z-scores may be below −2 and 2.5 % above 2. Properties: Univariate. It is unclear how to choose the width of the early and the late window?

Q3 How long do you need to monitor the chain to get results of sufficient MC accuracy? (< 0.001)
RELATED TO Q2 & Q3

Raftery & Lewis (Convergence to ergodic average) Idea: Run diagnostics based on a criterion of accuracy of estimation of the quantile q. It is a non-parametric approach. • Two state Markov-Chain theory, discretisation of a continuous chain to get a binary control $Z^{(t)} = I_{\theta^t \leq \theta_q}$ $P = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}$
• Pseudo-transition Matrix
• Needs a pre-run to set up P preliminary chain to estimate α, β
Q2 (Burn-in) (M) needed; Q3 Total N sample size N needed; N_{min} minimal sample needed $I = \frac{M+N}{N_{\min}}$ (independent),
The (in)dependence factor I, indicates to which extend autocorrelation inflates the required sample size. I > 5 strong autocorrelation. It is a crude estimate of the thinning interval. Quantiles which are closer to median need more N than quantiles further away.

Heidelberger & Welch (Convergence to stationarity)
Q2 Stationarity test: Basically a Kolmogorov-Smirnov Test. At each iteration 10% are removed of the first half of the chain. Information: if passed, iteration and p-value are provided. If all iterations have been used and did not pass, then failed.
Q3 When all passed then a Halfwidth test is applied on the retained chain.
95 % CI (mean) with portion of the chain which has passed stationarity test. Coefficient of variation = $\frac{S}{\mu}$ < eps = 0.1 desired accuracy, where S is half width of 95 % CI and μ is mean. If > eps then fails.
If the sample is not sufficient to estimate the mean with sufficient accuracy then increase it by 1.5.

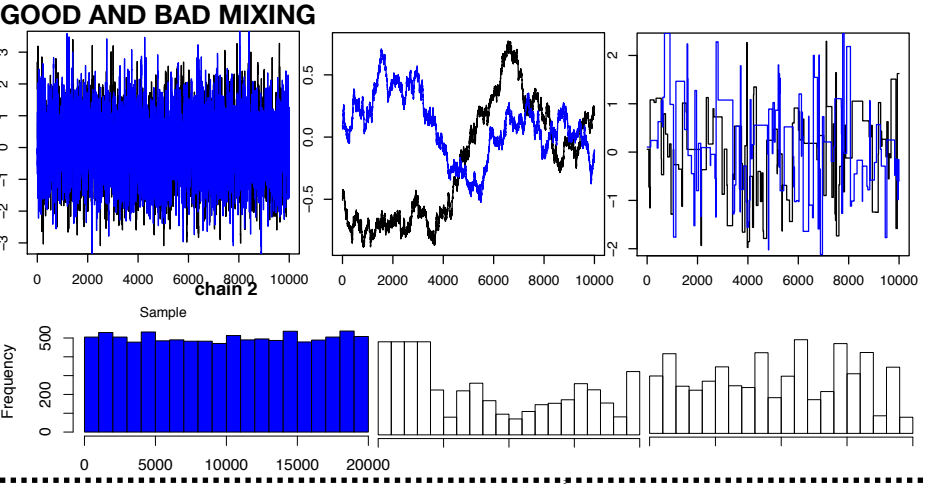
Why MCMC sampling can be dangerous?
Diagnostic tests are frequentist tests and there is a danger of false negative results, type II error β.

Truth, convergence to stationarity	Diagnostic test: non-significant	Diagnostic test: significant
yes	$1-\alpha$	α
no	β	$1-\beta$

		Convergence to stationarity	Convergence to ergodic average
number of chains 1	graphical	traceplots	NA
	numerical	H&W (run length control based on mean)	ESS
		Geweke (lack of convergence)	R&L
number of chains > 1	graphical	rank plots	NA
		NA	BGR, $\hat{R}$ (Lack of convergence using multiple parallel chains)

CODA (Overview)

GOOD AND BAD MIXING

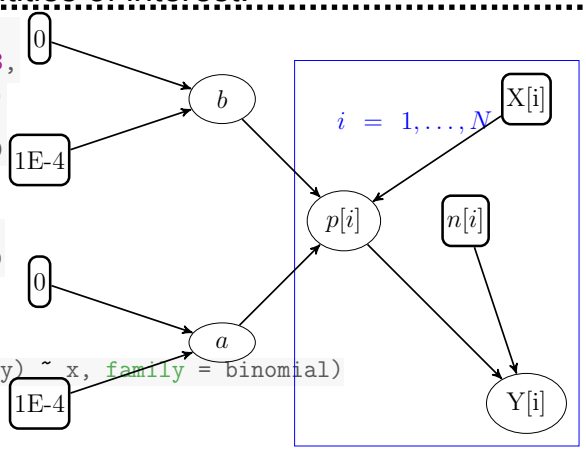


Left panel: good mixing obtained for a suitable standard deviation.Rank plot indicates uniformity for the chain. This shows that the chain has converged and is mixing well.
Middle panel: bad mixing obtained for a too small standard deviation.
Right panel: bad mixing obtained for a too large standard deviation.
ESS shows how many independent draws contain the same amount of information as the dependent sample obtained by the MCMC algorithm. The higher ESS the better. ESS must be large enough to get stable inferences for quantities of interest.

LOGISTIC REGRESSION IN RJAGS

```
#create Data
data <- matrix(NA, nrow=6, ncol=3)
colnames(data) <- list("x", "y", "n")
data[,1] <- x
data[,2] <- y
data[,3] <- n
mice.data <- list(y=data[,2], x=data[,1],
                  n=data[,3], N = nrow(data))
#define parameters
mice.params <- c("a", "b")
# define initial values for 2 chains
inits1 <- list(a=rnorm(1), b=rnorm(1),
               .RNG.name="base::Super-Duper", .RNG.seed=1)
inits2 <- list(a=rnorm(1), b=rnorm(1),
               .RNG.name="base::Wichmann-Hill", .RNG.seed=2)
mice.inits <- list(inits1, inits2)
modelString = "
# open quote for modelString
model{
  for(i in 1:N){
    y[i] ~ dbin(p[i],n[i])
    logit(p[i]) <- a+b*x[i]
  }
  a ~ dnorm(0,0.0001)
  b ~ dnorm(0,0.0001)
}" # close quote for modelString
writeLines(modelString, con="05_mice_logistic_regression_JAGS.txt")
# model initiation
mice.rjags <- jags.model(
  file = "05_mice_logistic_regression_JAGS.txt",
  data = mice.data,
  inits = mice.inits,
  n.chains = 2,
  n.adapt = 4000)
N.iter <- 10000
N.thin <- 1
N.burnin <- 4000
# burn-in
update(mice.rjags, n.iter = N.burnin)
# sampling/monitoring
fit.rjags.coda.mice <- coda.samples(
  model = mice.rjags,
  variable.names = mice.params,
  n.iter = N.iter,
  thin = N.thin)
# store samples for each parameter f
a.sample <- fit.rjags.coda.mice[,"a"]
b.sample <- fit.rjags.coda.mice[,"b"]
summary(fit.rjags.coda.mice)
```

```
# the covariate values (dose)
x_original <- c(0.0028, 0.0028,
   0.0056, 0.0112, 0.0225, 0.0450)
# center covariate values
x <- x_original - mean(x_original)
# number of mice deaths
y <- c(26, 9, 21, 9, 6, 1)
     # total number of mice
n <- c(28, 12, 40, 40, 40, 40)
## Fit a classical generalized linear model
   (logistic regression for binomial data)
m <- glm(formula = cbind(y, n - y) ~ x, family = binomial)
```

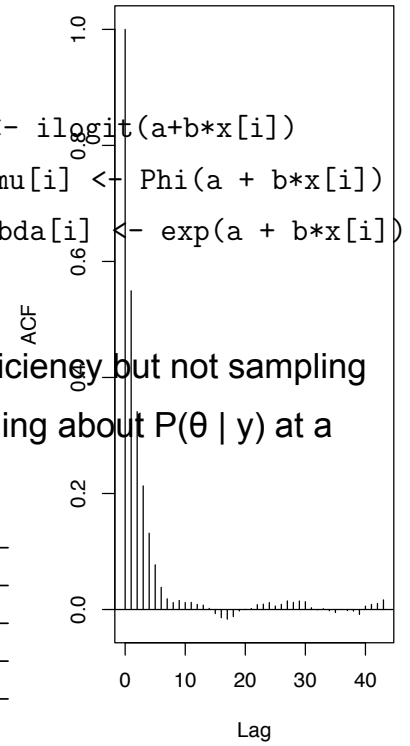


logit(p[i]) <- a+b*x[i] - binary - logit ↔ p[i] <- ilogit(a+b*x[i])

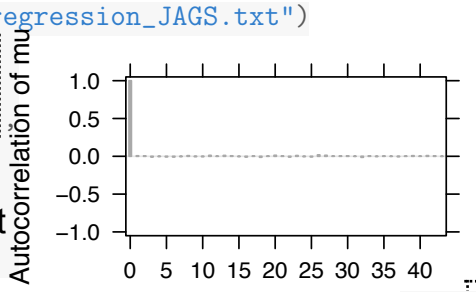
probit(mu[i]) <- a + b*x[i] - binary - probit ↔ mu[i] <- Phi(a + b*x[i])

log(lambda[i]) <- a+b*x[i] - Poisson - log ↔ lambda[i] <- exp(a + b*x[i])

```
a.sample <- fit.rjags.coda.mice[,"a"]
acf(as.matrix(a.sample), main = "a", ci = "")
```



ACF plots shows high dependence. –> Autocorrelation in the draws affects only efficiency but not sampling validity.
If autocorrelation is large, then we are learning about P(θ | y) at a slower rate than iid.



Autocorrelation plots give an overview of autocorrelation in different thinning interval. When the lag is larger than 1, the autocorrelation is close to 0.

```
heidel.diag(fit.rjags.coda.mice)
##   Stationarity start     p-value
##   test         iteration
## a passed       1         0.688
## b passed       1         0.442
##
##   Halfwidth Mean     Halfwidth
##   test
## a passed      -0.964 0.009
## b passed    -142.693 0.979
```

heidel.diag checks the convergence to stationarity and shows that the chains passed the stationarity and halfwidth

```
raftery.diag(fit.rjags.coda.mice)
Quantile (q) = 0.025
Accuracy (r) = +/- 0.005
Probability (s) = 0.95

   Burn-in  Total Lower bound  Dependence
   (M)      (N)   (Nmin)       factor (I)
 a 10       12690 3746         3.39
 b 12       13200 3746         3.52
```

3746: spl size per chain
n.thin*3750=n.iter

raftery.diag checks the convergence to ergodic average and suggests a thinning equal to 4 (maximum of all dependence

$$\mathrm{ACF}(k) = \mathrm{corr}(\theta_t^*, \theta_{t+k}^*)$$

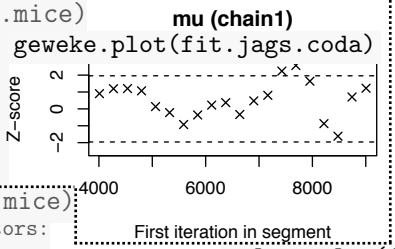
$$\mathrm{ESS} = N_{\mathrm{eff}} = \frac{M}{1 + 2\sum_{k>1} \mathrm{ACF}(k)}$$

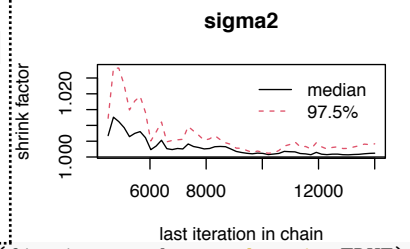
```
effectiveSize(fit.jags.coda)
```

geweke.diag and geweke.plot check the convergence to stationarity: geweke.diag shows that the z-scores are not larger than 1.96 and geweke.plot shows that only a few values fall out of the bound (-1.96, 1.96).

```
geweke.diag(fit.rjags.coda.mice)
Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

      a       b
-0.1719 -0.3351
```






```
gelman.plot(fit.jags.coda,autoburnin=TRUE)
```

gelman.diag and gelman.plot deal with the convergence to ergodic average
Figure demonstrates that the shrink factor ($\hat{R}$) decreases to 1 with increasing number of iterations.

stable.GR from package stableGR (Vats and Knudson). psrf is the "potential scale reduction factor" which is the R and these are smaller than 1.01.

```
gelman.diag(fit.rjags.coda.mice)
## Potential scale reduction factors:
##
##   Point est. Upper C.I.
## a          1          1
## b          1          1
##
## Multivariate psrf
##
## 1
```

```
stable.GR(fit.rjags.coda.mice)
## $psrf
## [1] 1.000062 1.000062
```

After an adaptation phase of 1000 and a burn-in of 4000 draw a sample of 1000 observations in one chain with thinning set to 1.

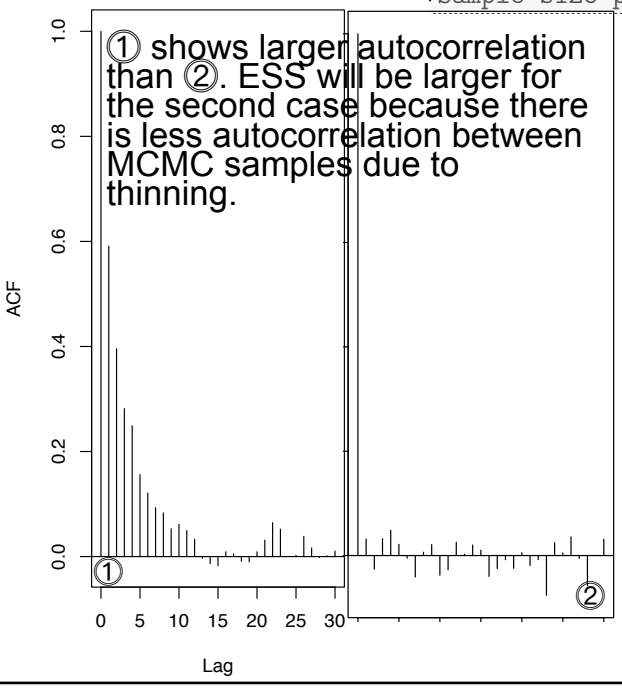
```
Iterations = 5001:6000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 1000
```

①

After an adaptation phase of 1000 and a burn-in of 4000 draw a sample of 10000 observations in one chain with thinning set to 10.

```
Iterations = 5010:15000
Thinning interval = 10
Number of chains = 1
Sample size per chain = 1000
```

②



① shows larger autocorrelation than ②. ESS will be larger for the second case because there is less autocorrelation between MCMC samples due to thinning.

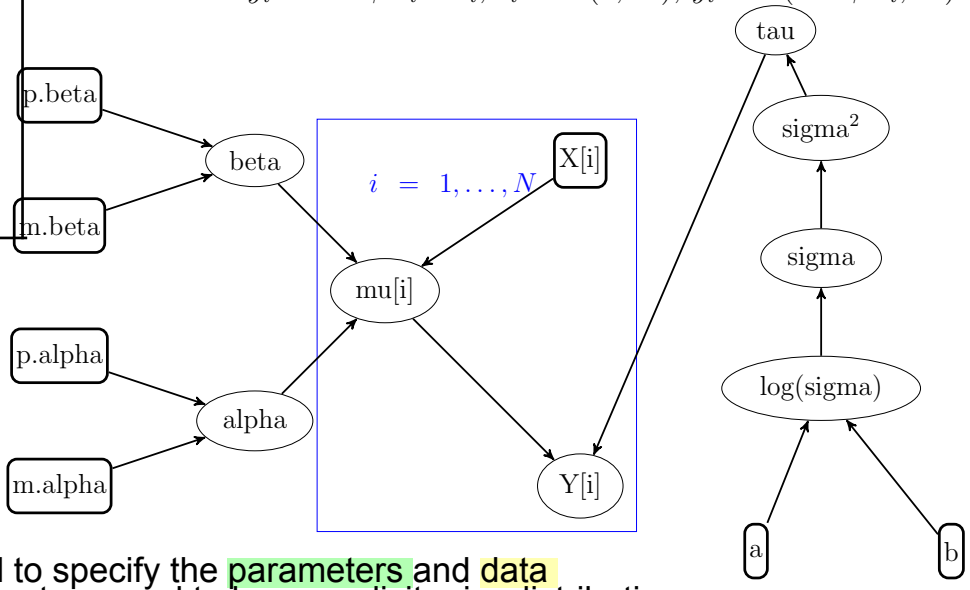
```
(n.eff(as.matrix(fit.rjags.coda.mice.1[, "b"])))
## $n.eff    $converged $n.target
##       se [1] FALSE      se
## 196.3447             31303
(n.eff(as.matrix(fit.rjags.coda.mice.2[, "b"])))
## $n.eff $converged $n.target
## var1   [1] FALSE  var1
## 1000             6147
```

n.eff from the R package stableGR informs about the convergence status.

BUGS CODE FOR A LINEAR REGRESSION MODEL

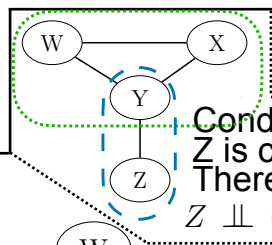
Remark: In R: `lm(y ~ x)`.

$$y_i = \alpha + \beta x_i + \epsilon_i,\ \epsilon_i \overset{iid}{\sim} \mathrm{N}(0, \sigma^2),\ y_i \sim \mathrm{N}(\alpha + \beta x_i, \sigma^2).$$




```
model {
# likelihood
for(i in 1:length(Y)) {
Y[i] ~ dnorm(mu[i], tau)
mu[i] <- alpha + beta * x[i]
}
# priors
alpha ~ dnorm(m.alpha, p.alpha)
beta ~ dnorm(m.beta, p.beta)
log.sigma ~ dunif(a,b)
sigma <- exp(log.sigma)
sigma.sq <- pow(sigma, 2)
tau <- 1/sigma.sq
}
```

- Need to specify the parameters and data
- parameters need to have explicit prior distributions
- not vectorized "for loops"
- model includes parameter transformations



Directed Acyclic Graph (DAG)
$p(W, X, Y, Z) = p(W)p(X)p(Y \mid W, X)p(Z \mid Y)$

Conditional Independence Graphs (CIG)
Z is conditionally independent of (W, X) given Y
There are two cliques in a CIG: (W, X, Y) and (Y, Z)
$Z \perp\!\!\!\perp (W, X) \mid Y$

(a) DAG (b) CIG

Figure 5.6: Marrying parent nodes to get CIG from a DAG