What's new in JAGS?

Martyn Plummer

International Agency for
Research on Cancer

AppliBUGS June 21 2018

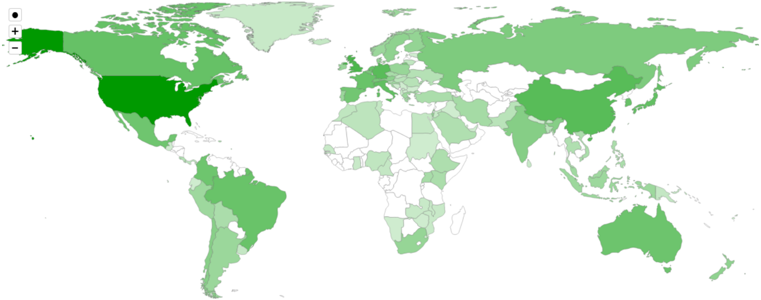
JAGS: Some statistics

JAGS 4.3.0 has been downloaded over 35,000 times from SourceForge since its release on 2017-08-10.

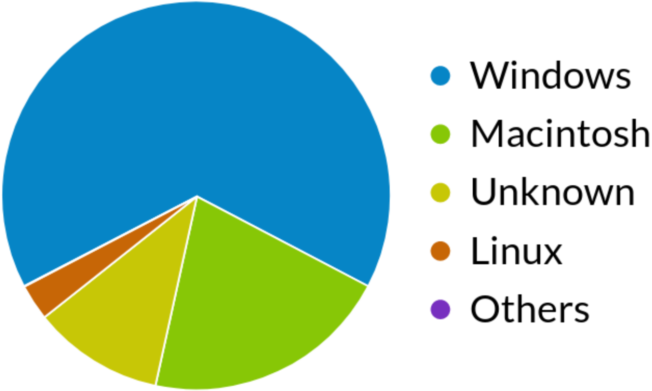
Platform	File	Downloads
Windows	JAGS-4.3.0.exe	22,951
MacOS	JAGS-4.3.0.dmg	8,365
Source	JAGS-4.3.0.tgar.gz	3,894

The user manual has been downloaded 4,792 times.

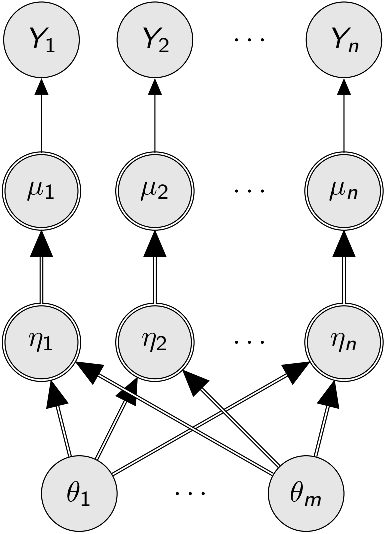
Top countries



Top operating systems



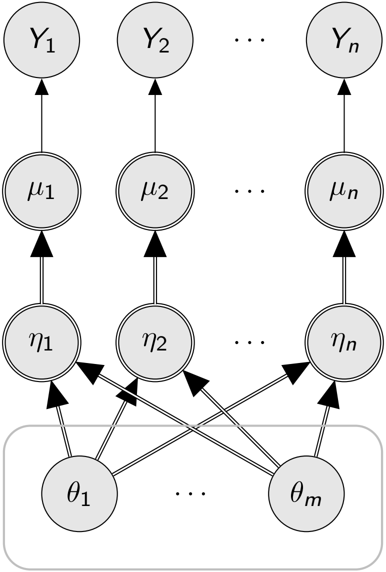
GLM as a design motif



A GLM is a sub-graph with the following elements

- **parameters** θ with prior normal distribution
- **linear predictors** η are linear functions of the parameters (intermediate nodes omitted).
- **link functions** transform linear predictor η to mean value μ
- **Outcome variables Y** depend on parameters θ via the mean μ

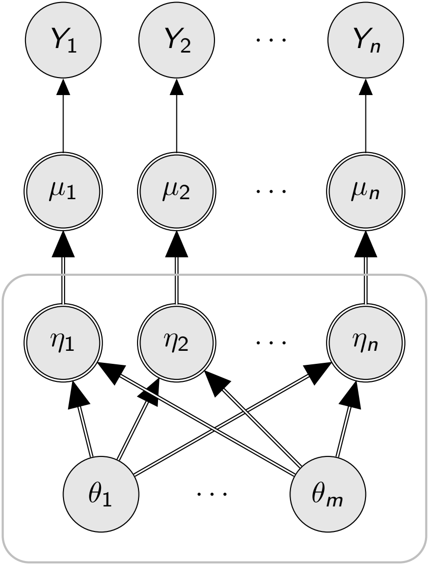
GLM as a design motif



A GLM is a sub-graph with the following elements

- **parameters $\boldsymbol{\theta}$** with prior normal distribution
- **linear predictors** η are linear functions of the parameters (intermediate nodes omitted).
- **link functions** transform linear predictor η to mean value μ
- **Outcome variables Y** depend on parameters $\boldsymbol{\theta}$ via the mean $\boldsymbol{\mu}$

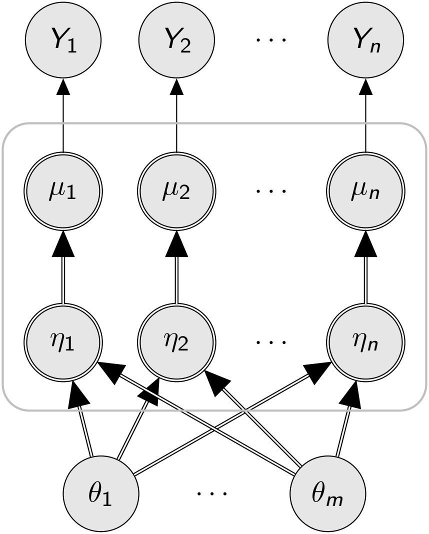
GLM as a design motif



A GLM is a sub-graph with the following elements

- **parameters** $\boldsymbol{\theta}$ with prior normal distribution
- **linear predictors** $\boldsymbol{\eta}$ are linear functions of the parameters (intermediate nodes omitted).
- **link functions** transform linear predictor $\boldsymbol{\eta}$ to mean value $\boldsymbol{\mu}$
- **Outcome variables Y** depend on parameters $\boldsymbol{\theta}$ via the mean $\boldsymbol{\mu}$

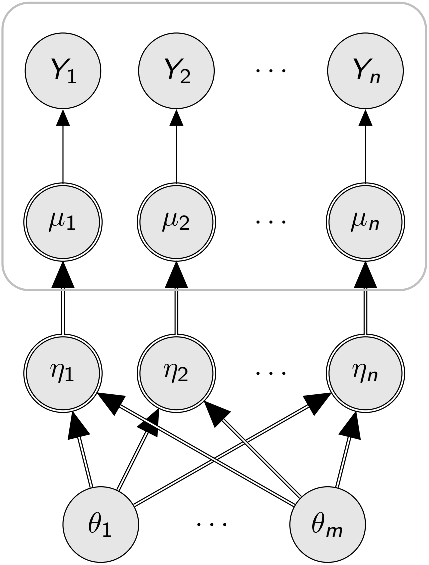
GLM as a design motif



A GLM is a sub-graph with the following elements

- **parameters** $\boldsymbol{\theta}$ with prior normal distribution
- **linear predictors** $\boldsymbol{\eta}$ are linear functions of the parameters (intermediate nodes omitted).
- **link functions** transform linear predictor $\boldsymbol{\eta}$ to mean value $\boldsymbol{\mu}$
- **Outcome variables Y** depend on parameters $\boldsymbol{\theta}$ via the mean $\boldsymbol{\mu}$

GLM as a design motif



A GLM is a sub-graph with the following elements

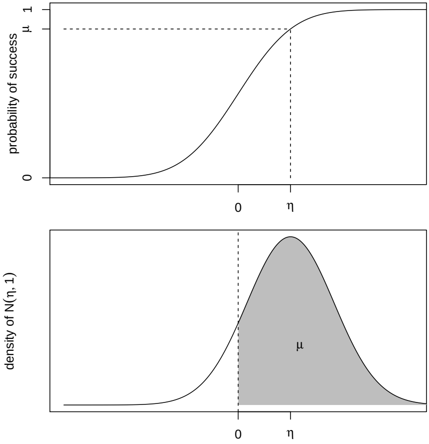
- **parameters** $\boldsymbol{\theta}$ with prior normal distribution
- **linear predictors** $\boldsymbol{\eta}$ are linear functions of the parameters (intermediate nodes omitted).
- **link functions** transform linear predictor $\boldsymbol{\eta}$ to mean value $\boldsymbol{\mu}$
- **Outcome variables Y** depend on parameters $\boldsymbol{\theta}$ via the mean $\boldsymbol{\mu}$

Auxiliary sampling for GLMs

Multiple data augmentation methods have been proposed

- It seems natural to preserve the benefits of the linear sampler by extending its scope.
- This also has the benefit of code reuse as a single sampling "engine" can address multiple models
- Some GLMs can be reduced to linear form by data augmentation (adding additional nodes to the graph)
- Methods have been proposed for Poisson regression and logistic regression, which are coincidentally the most common models in epidemiology

Albert and Chib (1993) approach to binary probit models



$$\mu \equiv P(Y = 1 \mid \eta) = \Phi(\eta)$$

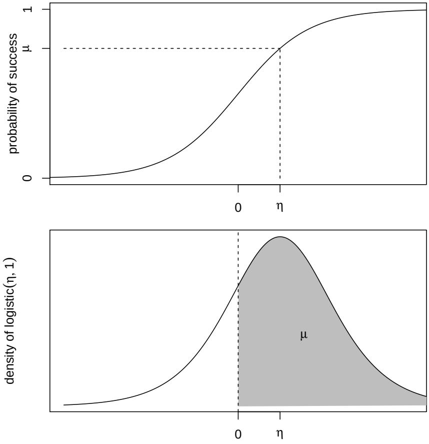
Albert and Chib (1993) introduce a latent variable

$$Z \sim N(\eta, 1)$$

and make the outcome Y a deterministic function of Z

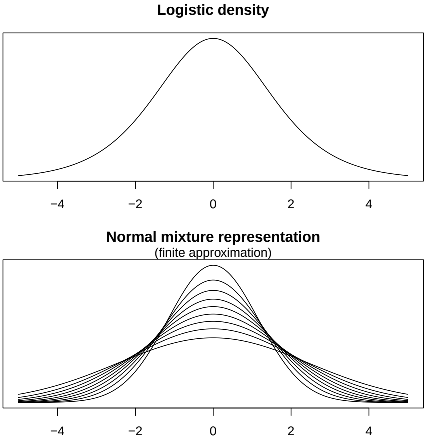
$$Y = I\{Z \geq 0\}$$

Holmes and Held (2006) approach to binary logit models



Logistic regression models with a binary outcome also have a latent variable representation, where the latent Z has a logistic distribution.

Mixture representation of logistic distribution

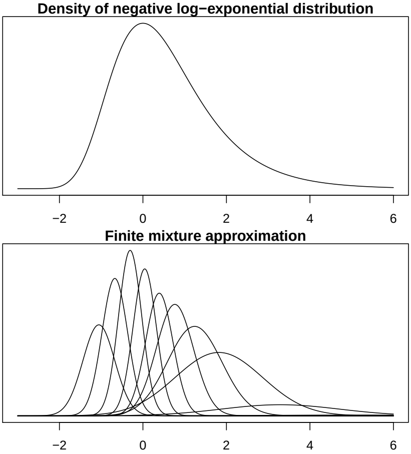

Logistic density

Normal mixture representation
(finite approximation)

The logistic distribution is a scale mixture of normals, where the scale parameter has a Kolmogorov-Smirnov distribution

$$\begin{array}{rcl} Z \mid \psi & \sim & N(0,(2\psi)^2) \\ \psi & \sim & KS \end{array}$$

Frühwirth-Schnatter et al (2010): Poisson and logistic regression

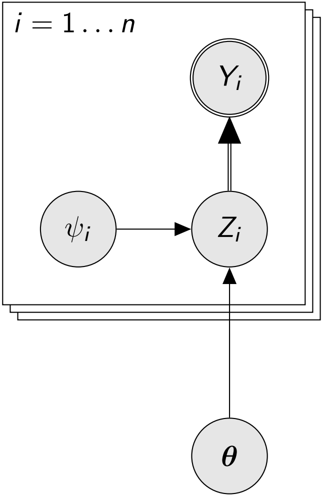


Latent negative log-gamma variables with integer shape parameter are approximated by finite mixtures of normal distributions.

Graphical representation

All these data augmentation schemes have the same graphical representation.

- Z_i is the latent variable used when updating $\boldsymbol{\theta}$.
- ψ_i is a mixture parameter that determines which normal approximation is used
- $(Z_i, \psi_i) \mid (Y_i, \boldsymbol{\theta})$ can be easily sampled
- $Z_i \mid (\psi_i \boldsymbol{\theta})$ appears normal



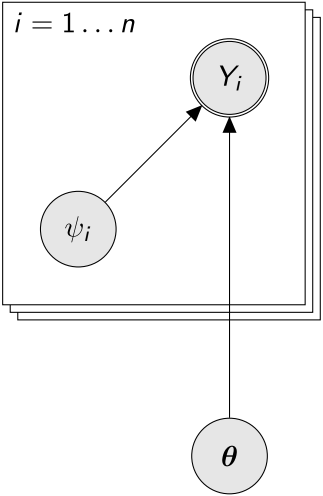
Polson, Scott and Windle (2012): binomial logistic models

A pseudo-likelihood approach. The logistic likelihood (i.e. binomial and negative binomial distributions) can be represented as a Polya-gamma mixture of normals

$$\frac{(\exp(\eta))^a}{(1+\exp(\eta))^b} = 2^{-b}\exp(\kappa\eta)\int_0^\infty \exp(-\omega\eta^2/2)p(\omega)d\omega$$

where $\kappa = a - b/2$ and $\omega \sim PG(b, 0)$

- Polson, Scott and Windle (2012) scheme represents likelihood of Y_i as a scale mixture of normals, without any latent outcome variable Z_i representation.
- Hence very efficient sampling without need to resample η_i, Z_i



Ordered probit and logit

Generalizations of the binary probit and logistic models to ordered outcomes:

```
Y[i] ~ dordered.logit(eta[i], cutpoints[1:Ncut])
Z[i] ~ dordered.probit(eta[i], cutpoints[1:Ncut])
```

This also fits into the GLM machinery via data augmentation.

Better initialization for observable functions

- Observable functions impose *a posteriori* constraints
  `Y ~ dinterval(t, C[1:m])` $C_y < t \leq C_{y+1}$
  `Y ~ sum(x1[1:m1], x2[1:m2],...)` $Y = \sum_{i=1} \sum_{j=1}^{m_i} x_{ij}$
- User must supply initial values for arguments
- The sample method for the sum observable function automatically fixes up initial values of `x1[1:m1]`, `x2[1:m2]`, ...

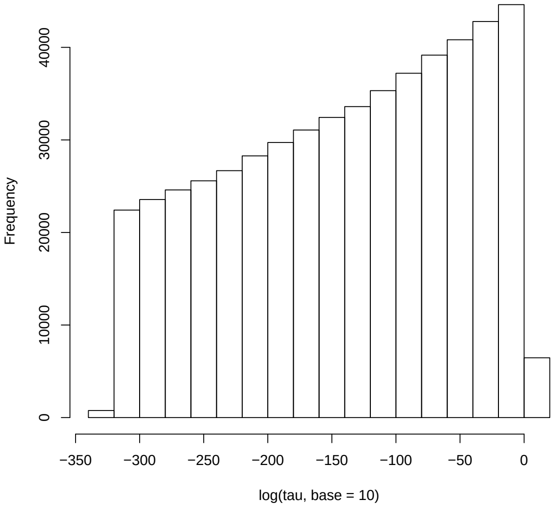
Initialization

- JAGS $\leq$ 4.3.0 sets initial values from a "typical value" (prior mean, median, mode, ...)
  - Problem: Parallel chains not independent, initial values not over-dispersed
- JAGS 5.0.0 draws initial values from the prior distribution.
  - Problem: Diffuse gamma prior

    ```
    tau ~ dgamma(1.0E-3, 1.0E-3)
    ```

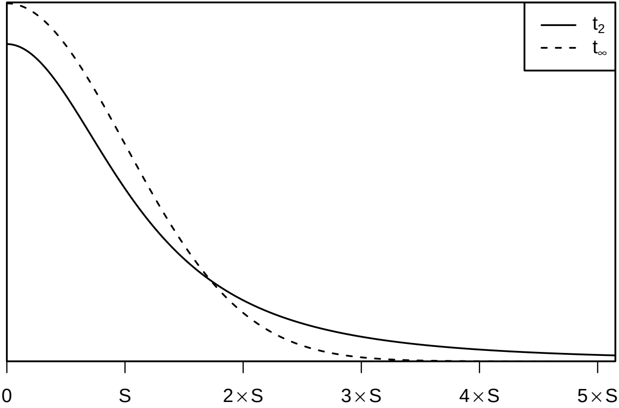
    for precision parameters never gives good initial values

1 million samples from the diffuse gamma prior



The scaled half-t prior

Gelman (2006) proposed the scaled half-t prior for the standard deviation of random effects.



But in the BUGS language, the normal distribution is parameterized by its

The scaled.gamma prior

The glm module provides the `scaled.gamma` distribution

```
tau ~ dscaled.gamma(S, df)
sigma <- 1/sqrt(tau)
```

Then sigma has half-t distribution with scale `S` and `df` degrees of freedom.

The scaled.wishart prior

Multivariate generalization due to Huang and Wand (2013)

```
Tau[1:m,1:m] ~ dscaled.wishart(S[1:m], df)
X[1:m] ~ dnorm(rep(0,m), Tau[1:m,1:m])
```

- Standard deviation of `X[i]` has half-t prior with scale `S[i]` and `df` degrees of freedom.
- If `df = 2` then the correlation between `X[i]` and `X[j]` for $j \neq i$ has uniform prior on $(-1, 1)$.

Efficient sampling of variance components

- As noted by Gelman (2006) the scaled half-t distribution has a redundant parameterization that allows efficient sampling.
- This strategy generalizes to the scaled Wishart distribution.
- JAGS uses efficient samplers when `scaled.gamma` or `scaled.wishart` is used for the prior precision of random effects in a GLM.
- Both special cases of the ancillary-sufficiency interleaving strategy (ASIS) of Yu and Meng (2011).

ASIS

- Ancillary parameterization:

$$\begin{aligned} \eta_i &= \alpha + \beta x + \sigma\epsilon_i \\ \epsilon_i &\sim N(0,1) \end{aligned}$$

- Sufficient parameterization:

$$\begin{aligned} \eta_i &= \alpha + \beta z + \xi_i \\ \xi_i &\sim N(0,\sigma^2) \end{aligned}$$

These are equivalent with $\xi_i = \sigma\epsilon_i$ but Gibbs sampling on σ is very different.

Interleaving both parameterizations gives "the best of both worlds´´.

Parallelism

In JAGS 5.0.0 OpenMP parallelism allows parallel chains to run on separate cores.

```
for (unsigned int iter = 0; iter < niter; ++iter) {

  #pragma omp parallel for num_threads(_nchain)
  for (unsigned int n = 0; n < _nchain; ++n) {
    for (auto i = _samplers.begin(); i != _samplers.end(); ++i)
      (*i)->update(n, _rng[n]);
    }
    ...
  }
}
```

Much more parallelisation is possible. See Goudie et al, MultiBUGS: Massively parallel MCMC for Bayesian hierarchical models
https://arxiv.org/abs/1704.03216

Issues with parallelism

A lot of code lies in the *dynamic scope* of the parallel for loop.

```
static int ngam = 0;
static double xmin = 0, xmax = 0., xsml = 0., dxrel = 0.;

#pragma omp threadprivate(ngam, xmin, xmax, xsml, dxrel)
```

All code in dynamic scope must be thread safe.

Compiler improvements

In JAGS $\leq$ 4.3.0, the compiler will not accept this:

```
n <- 10
for (i in 1:n) {
y[i] ~ dnorm(mu[i], tau)
...
}
```

This is now possible in JAGS 5.0.0.

Summary

- Expanding facilities of the glm module
  - Block sampling of fixed and random effects
  - Better prior distributions for variance components
  - More efficient sampling of variance components
- Changes to initialization
- Parallel chains
  - More parallelism in development