# Sampling methods for generalized linear models

Martyn Plummer

International Agency for
Research on Cancer

4 May 2012

# BUGS

- BUGS stands for Bayesian inference Using Gibbs Sampling.
- Starting in the late 1980s, the BUGS project brought together:
  1. Bayesian Inference
  2. Graphical modelling
  3. Simulation-based inference

  To provide user-friendly software for Bayesian data analysis.
- Growth in computing power has made simulation-based methods increasingly accessible.
- BUGS has helped to popularize Bayesian methods, which previously had limited applicability.

# Other Bayesian software

| | |
|---|---|
| PyMC | MCMC for Python |
| | http://code.google.com/p/pymc |
| HBC | Hierarchical Bayes Compiler |
| | http://www.cs.utah.edu/~hal/HBC |
| YADAS | Yet Another Data Analysis System |
| | http://www.stat.lanl.gov/yadas |
| HYDRA | MCMC library |
| | http://sourceforge.net/projects/hydra-mcmc |
| Scythe | Statistical library (MCMCpack) |
| | http://scythe.wustl.edu |
| CppBUGS | C++ version of BUGS |
| | https://github.com/armstrtw/CppBugs |
| Stan | A C++ library for probability and sampling |
| | http://code.google.com/p/stan/ |

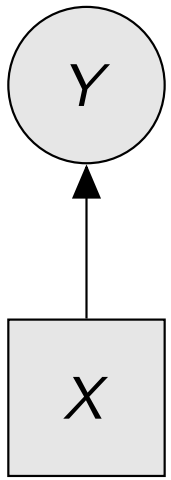
# JAGS: Just Another Gibbs Sampler

- A clone of BUGS
- Written in C++
  - Cross-platform (Linux, Windows, Mac OS X, ...)
- Object-based R interface (rjags)
  - Other interfaces: R2jags, runjags

JAGS aims (more or less) for compatibility with BUGS. In particular, models are described in the same way using "the BUGS language"

# Statistical models as graphs

In a graphical model, random variables are represented as nodes, and the relations between them by edges.
In simple models, we want to predict a single variable $Y$ from input variables $X$. This can be represented by a trivial graph:




Graphical models become more interesting when we have multiple variables, and the relations between them become more complex.

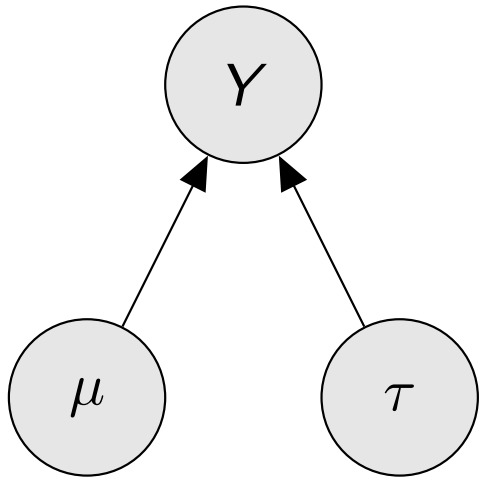
# Stochastic Relations

The relation

$$Y \sim N(\mu, \tau^{-1})$$

is written as

```
Y ~ dnorm(mu, tau)
```




This relation can be represented by a graph in which $Y$, $\mu$, $\tau$ are nodes. The dependency of $Y$ on parameters $\mu$, $\tau$ is represented by directed edges.

# Stochastic relations

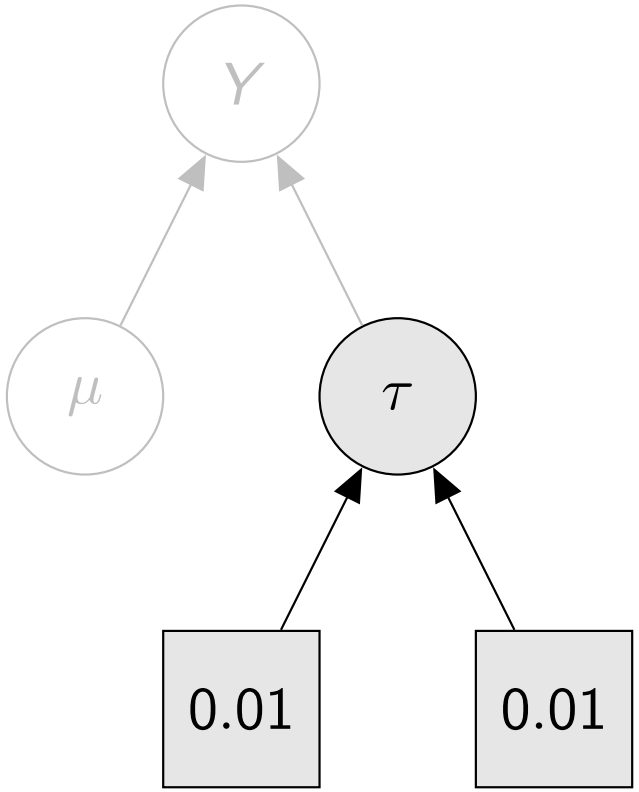
A parameter can itself have a distribution with its own hyper-parameters

$$\tau \sim \Gamma(0.01, 0.01)$$

In the BUGS language

```
tau ~ dgamma(0.01, 0.01)
```

This fits with the Bayesian approach to statistical inference, in which the parameters of a model are also random variables.



# Deterministic relations

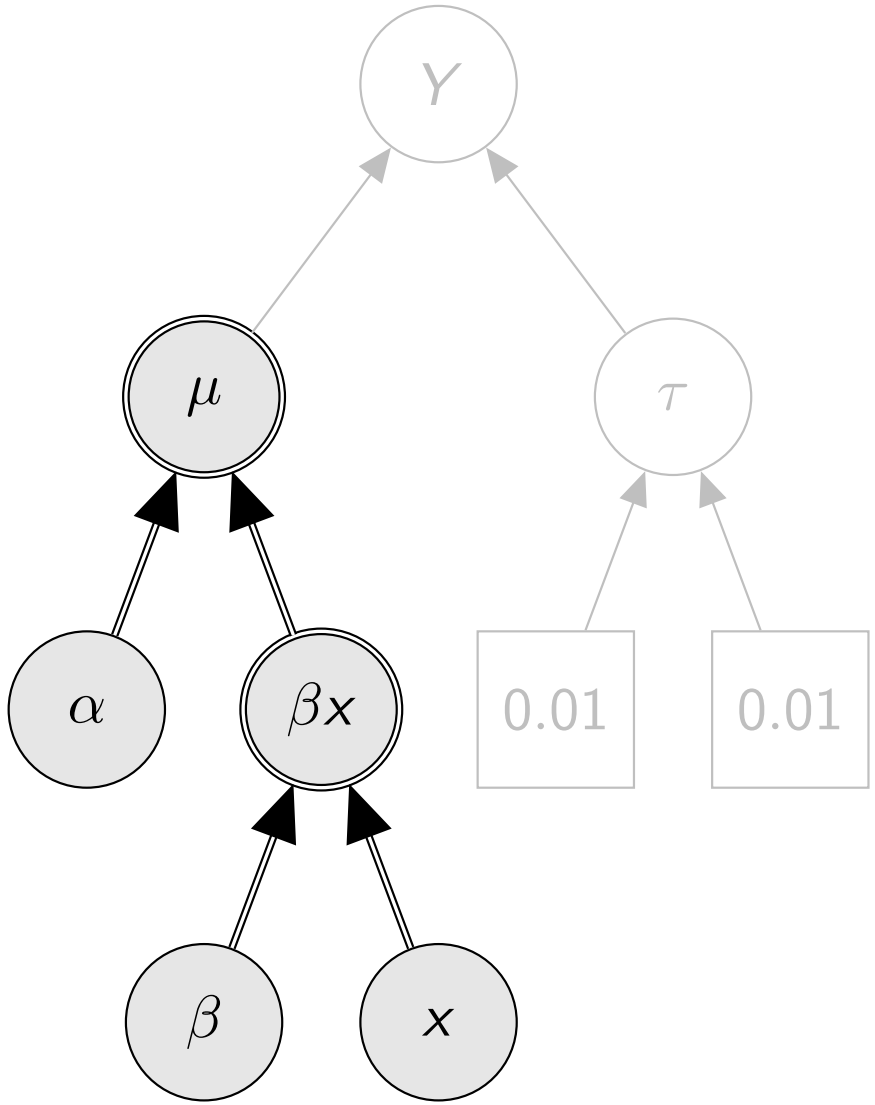
We can also describe deterministic relationships between variables

$$\mu = \alpha + \beta x$$

In BUGS:

```
mu <- alpha + beta * x
```

They are represented by double arrows.



# Arrays and for loops

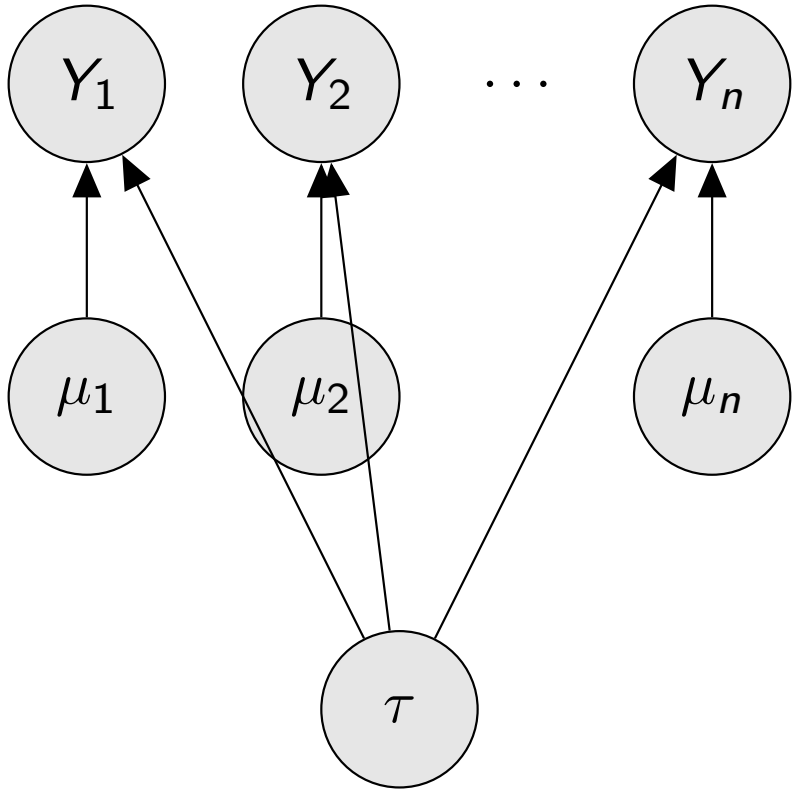
Repeated structures in the graph can be simplified using arrays and for loops.

$$Y_i \sim N(\mu_i, \tau^{-1}) \quad i = 1 \ldots n$$

In BUGS:

```
for (i in 1:n) {
   Y[i] ~ dnorm(mu[i], tau)
}
```

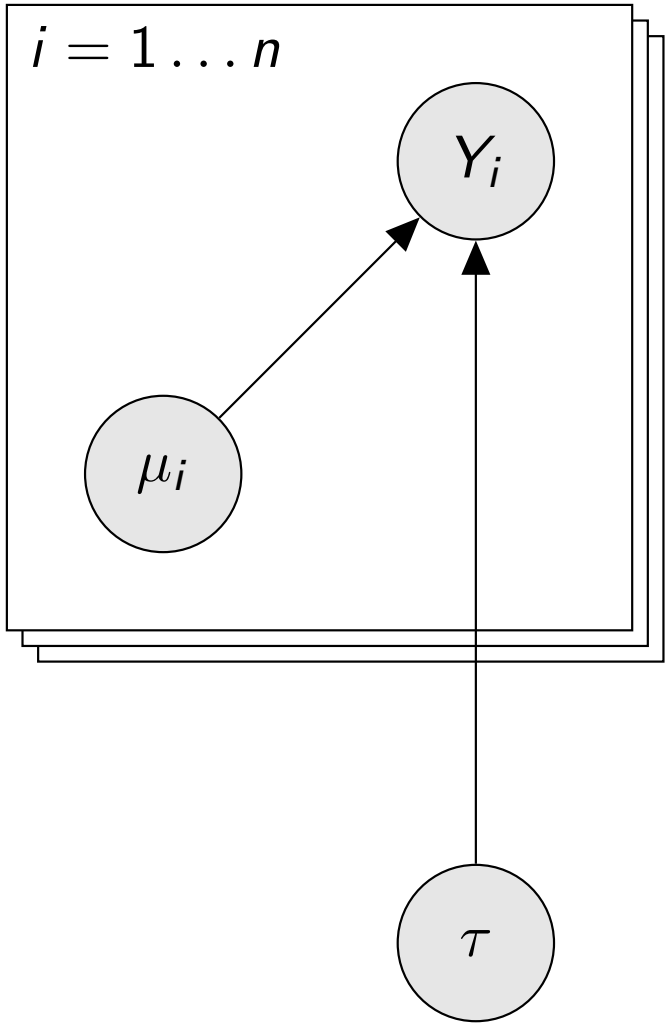
Here the nodes `Y[1]` to `Y[n]` are embedded in the array `Y`. Matrices and higher-dimensional arrays can also be used.



# Plates

Repeated structures can make the graph hard to read.
To simplify drawing of the graph, we use a "plate" notation.

- Only one entry in the for loop is shown.
- The rest are implied by the stack of plates



# Linear regression example (in BUGS)

Models written in the BUGS language may appear quite verbose:

```
for (i in 1:N) {
   y[i] ~ dnorm(mu[i], tau)
   mu[i] <- alpha + beta * x[i]
}
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
```

- You need to specify the *parameters* as well as the data.
- Parameters need to have explicit *prior distributions*
- Unlike R, the language is not vectorized, so you need `for` loops for repeated calculations.
- The model may include parameter transformations.

# Linear regression example (in BUGS)

Models written in the BUGS language may appear quite verbose:

```
for (i in 1:N) {
   y[i] ~ dnorm(mu[i], tau)
   mu[i] <- alpha + beta * x[i]
}
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
```

- You need to specify the *parameters* as well as the data.
- Parameters need to have explicit *prior distributions*
- Unlike R, the language is not vectorized, so you need `for` loops for repeated calculations.
- The model may include parameter transformations.

# Linear regression example (in BUGS)

Models written in the BUGS language may appear quite verbose:

```
for (i in 1:N) {
   y[i] ~ dnorm(mu[i], tau)
   mu[i] <- alpha + beta * x[i]
}
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
```

- You need to specify the *parameters* as well as the data.
- Parameters need to have explicit *prior distributions*
- Unlike R, the language is not vectorized, so you need `for` loops for repeated calculations.
- The model may include parameter transformations.

# Linear regression example (in BUGS)

Models written in the BUGS language may appear quite verbose:

```
for (i in 1:N) {
   y[i] ~ dnorm(mu[i], tau)
   mu[i] <- alpha + beta * x[i]
}
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
```

- You need to specify the *parameters* as well as the data.
- Parameters need to have explicit *prior distributions*
- Unlike R, the language is not vectorized, so you need `for` loops for repeated calculations.
- The model may include parameter transformations.

# Linear regression example (in BUGS)

Models written in the BUGS language may appear quite verbose:

```
for (i in 1:N) {
   y[i] ~ dnorm(mu[i], tau)
   mu[i] <- alpha + beta * x[i]
}
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
```

- You need to specify the *parameters* as well as the data.
- Parameters need to have explicit *prior distributions*
- Unlike R, the language is not vectorized, so you need `for` loops for repeated calculations.
- The model may include parameter transformations.

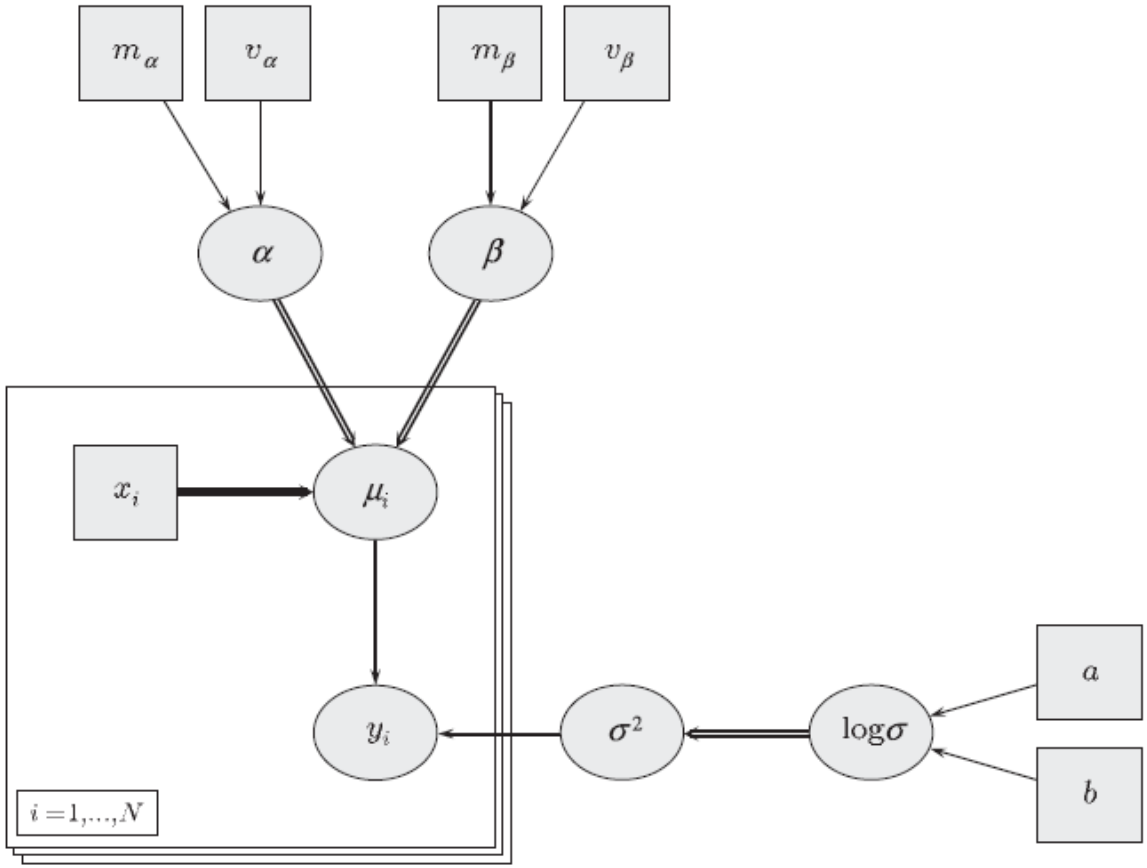
# Linear regression example (in BUGS)

Models written in the BUGS language may appear quite verbose:

```
for (i in 1:N) {
   y[i] ~ dnorm(mu[i], tau)
   mu[i] <- alpha + beta * x[i]
}
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
```

- You need to specify the *parameters* as well as the data.
- Parameters need to have explicit *prior distributions*
- Unlike R, the language is not vectorized, so you need `for` loops for repeated calculations.
- The model may include parameter transformations.

# A linear regression example



# Markov Chain Monte Carlo

A general purpose technique for simulation from an arbitrary distribution

- Usually abbreviated MCMC
- Developed in statistical physics in the 1950s
- Introduced to statistics via image analysis in the 1980s
- There is a catch!
  - The samples are *dependent*
  - The samples only come from the target distribution *in the long run*
  - But you cannot tell how long the "long run" should be! (convergence diagnostics)

# Gibbs sampling on a graphical model

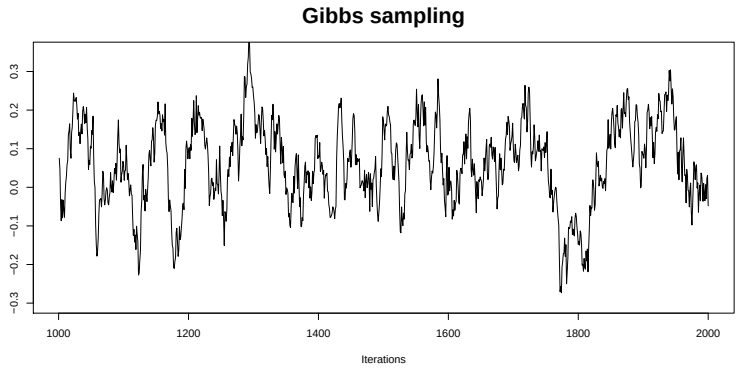
- A BUGS model defines a distribution on a set of nodes $(v_1, \ldots v_n)$ on a graph $G$.
- The distribution factorizes as

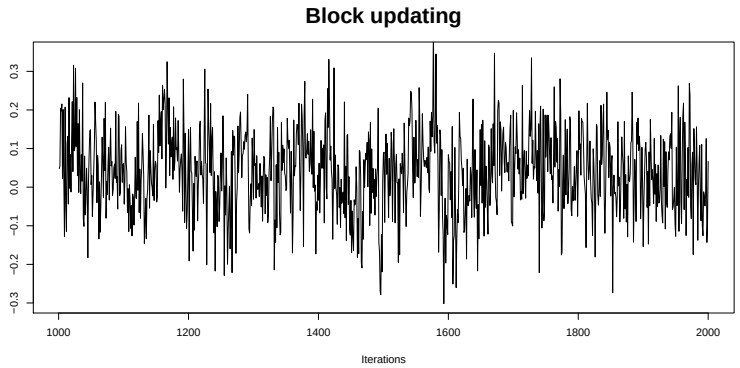
$$p(\mathbf{v}) = \prod_{i \in G} p(v_i \mid \text{Parents}(v_i))$$

- "Gibbs sampling" consists of visiting each node in turn, updating it in a way that preserves the joint distribution of $\mathbf{v}$
- The calculations required are local computations on the graph

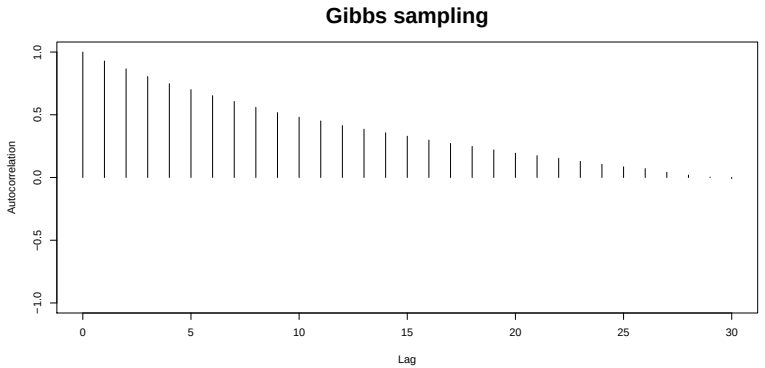
# Good and bad "mixing"

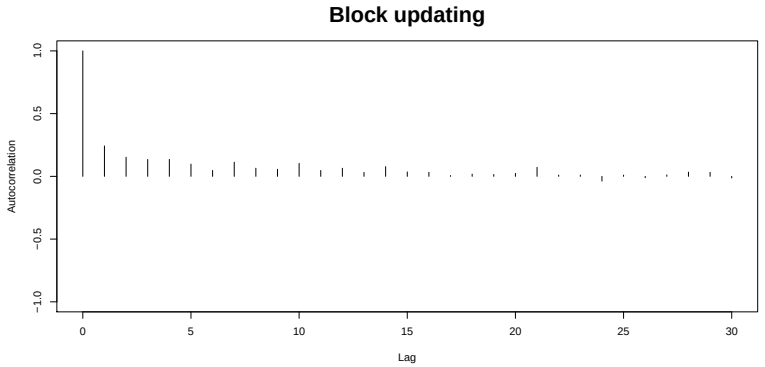
Trace plots show the evolution of the sampled values by the number of iterations.



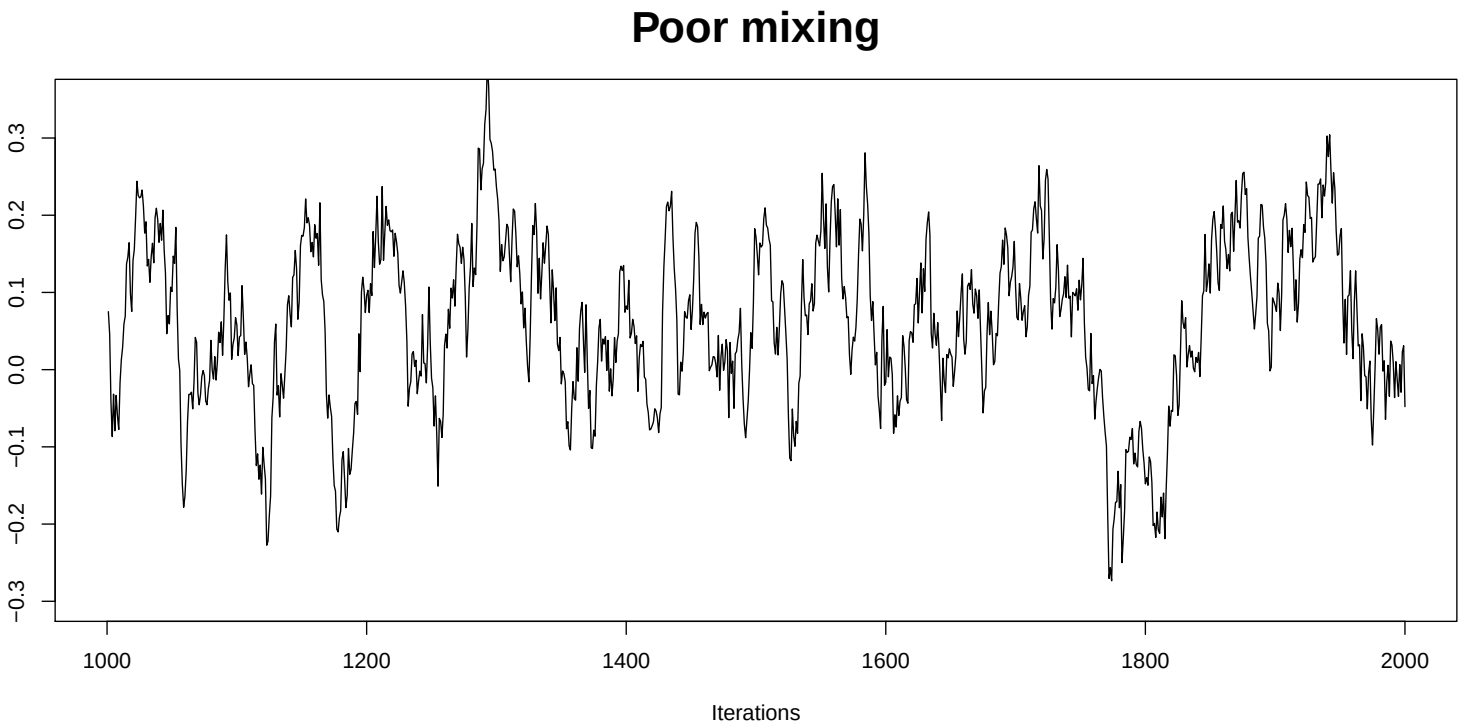




Autocorrelation plots show the extent to which current value depends on previous ones.

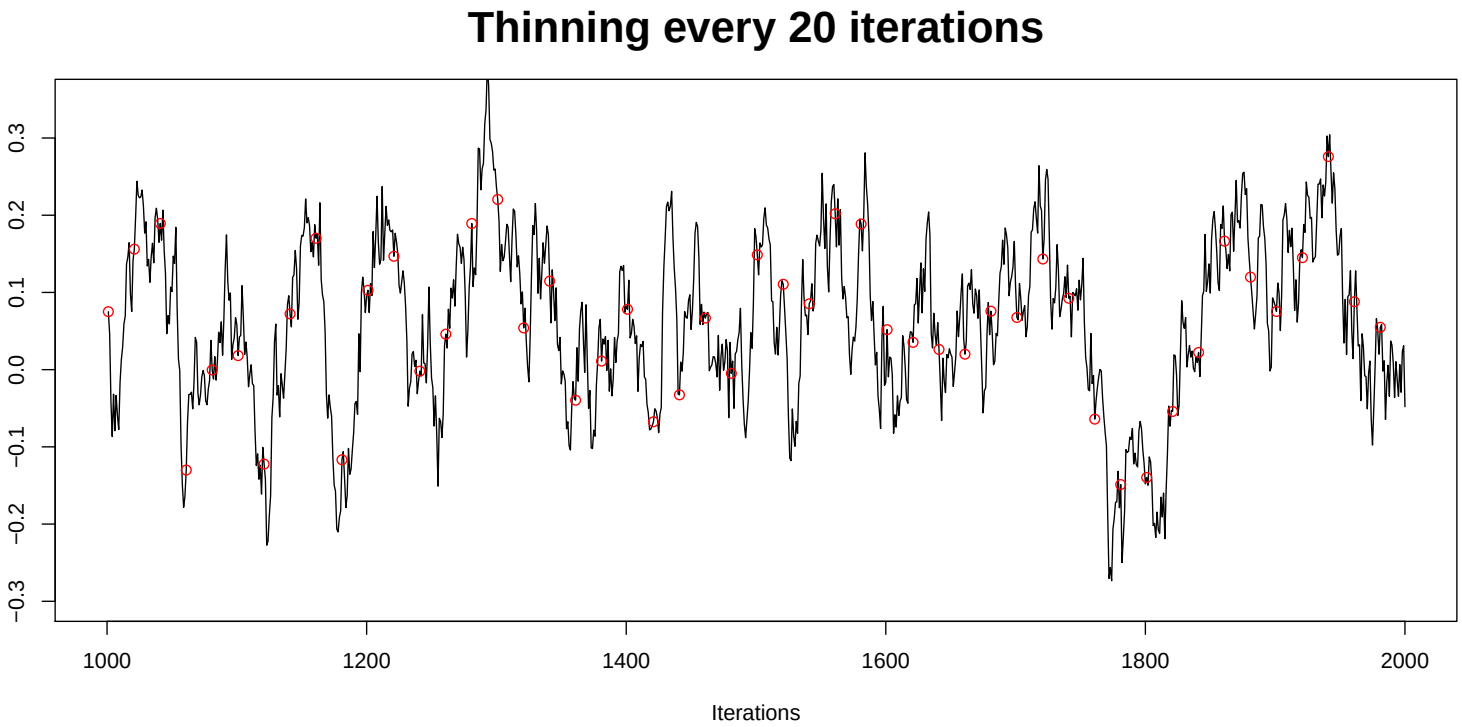


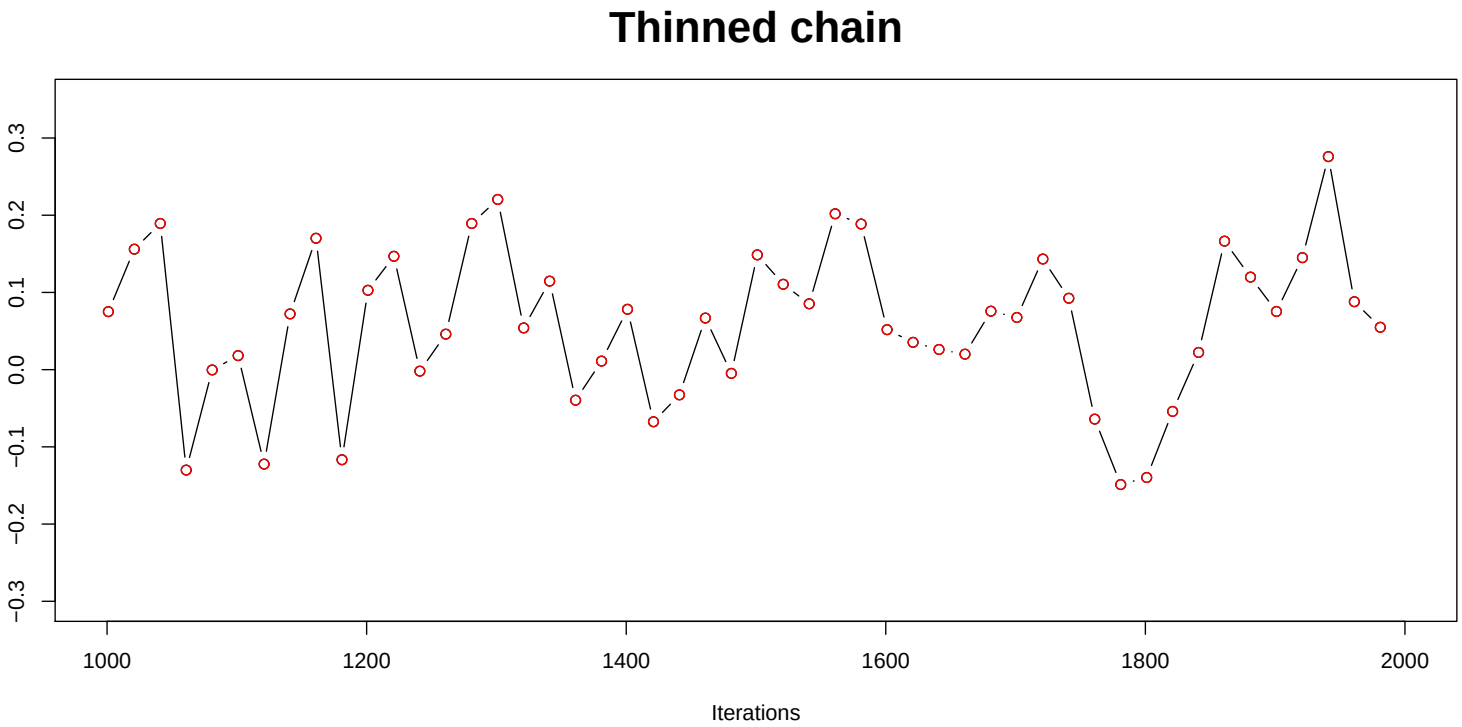
# "Solving" autocorrelation by thinning



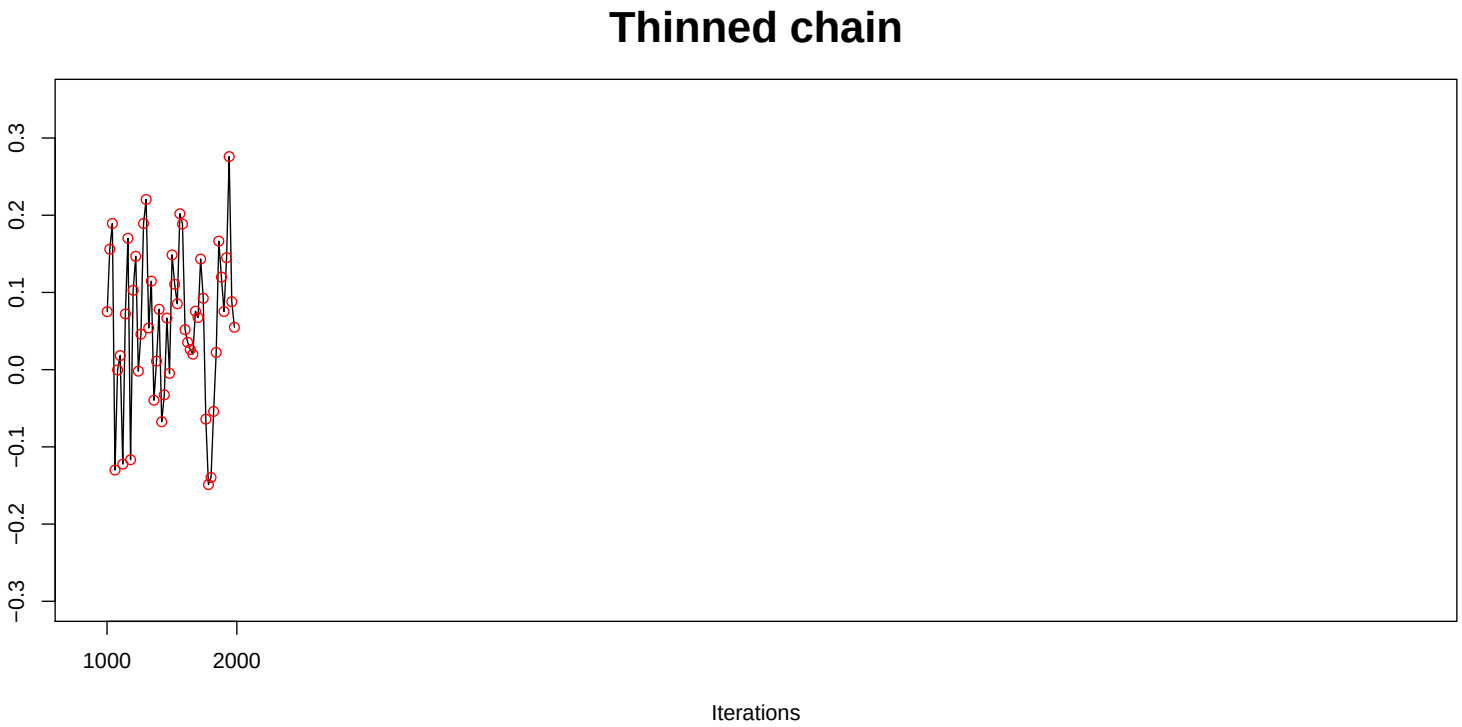
# "Solving" autocorrelation by thinning



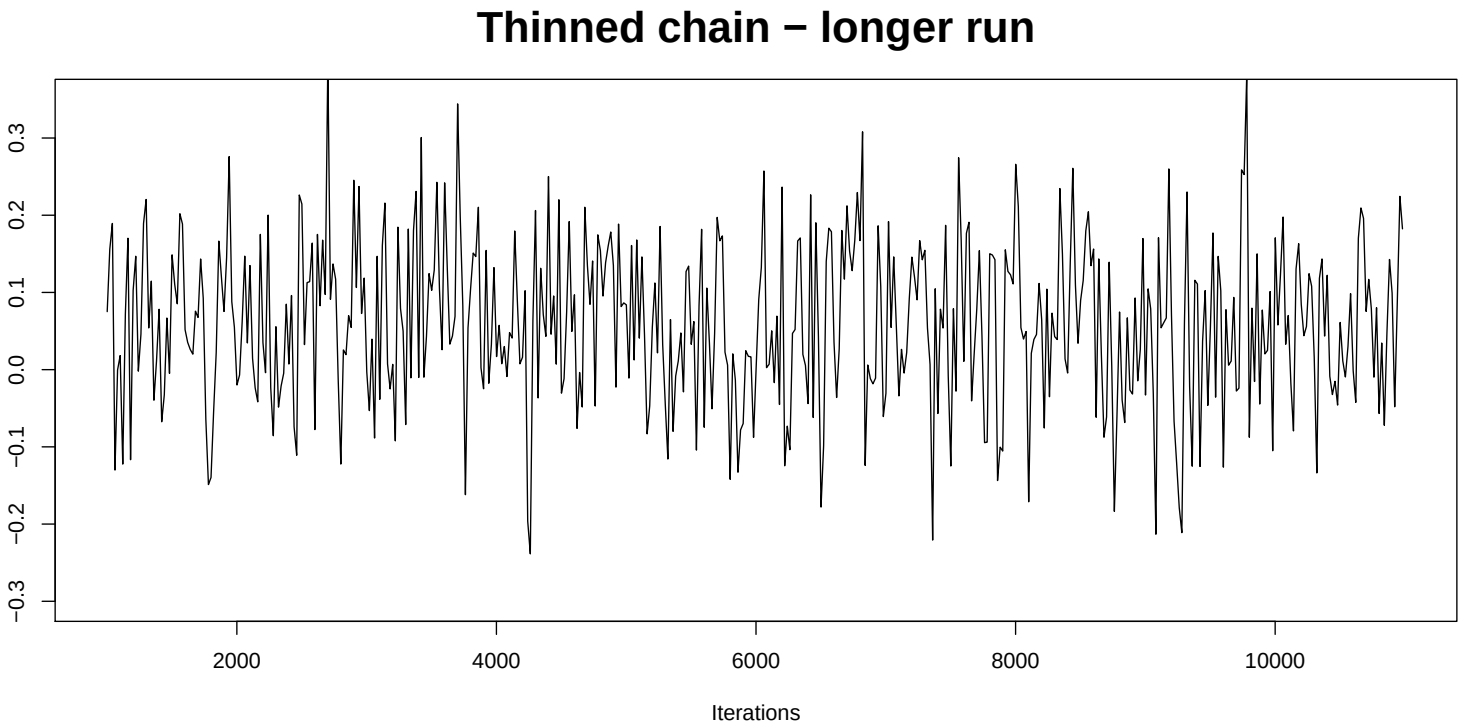
# "Solving" autocorrelation by thinning



# "Solving" autocorrelation by thinning

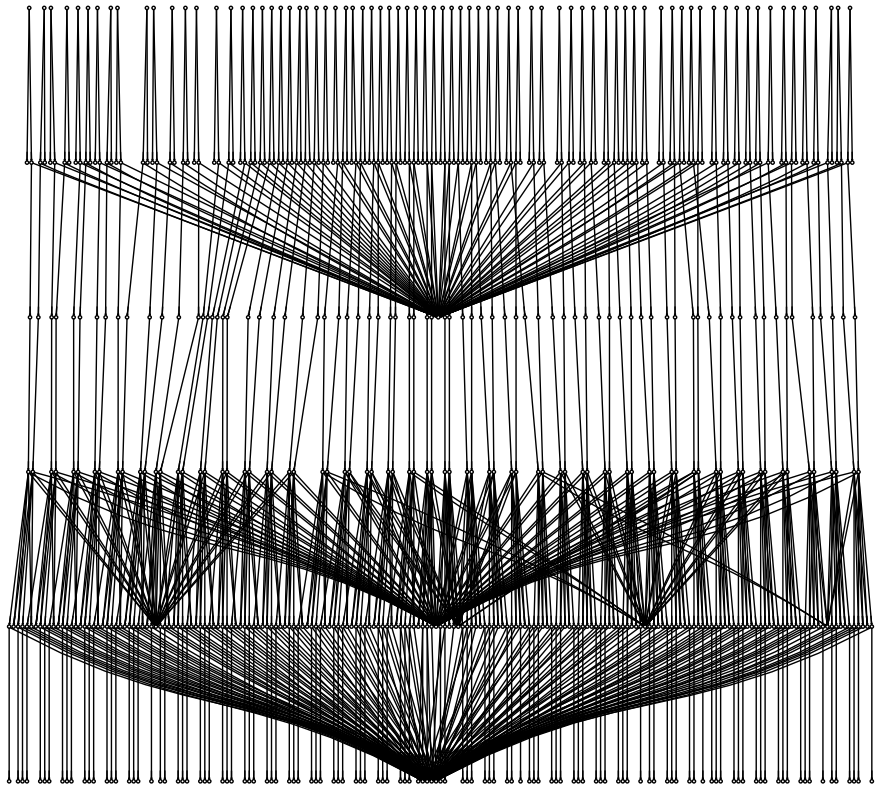


# "Solving" autocorrelation by thinning

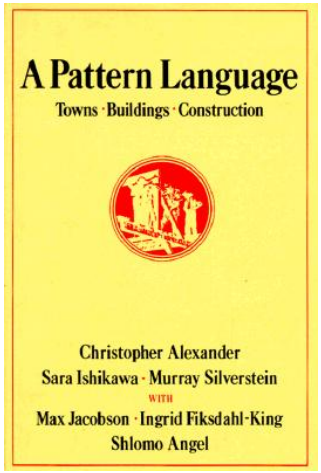


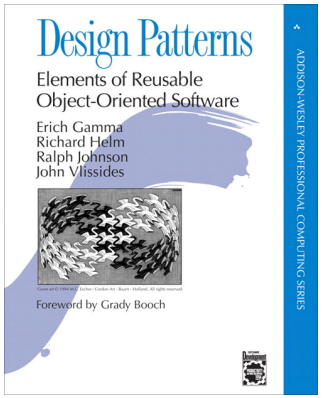
# An abstract view of the KIDNEY model

This is what a right-censored survival analysis problem looks like as a graphical model.



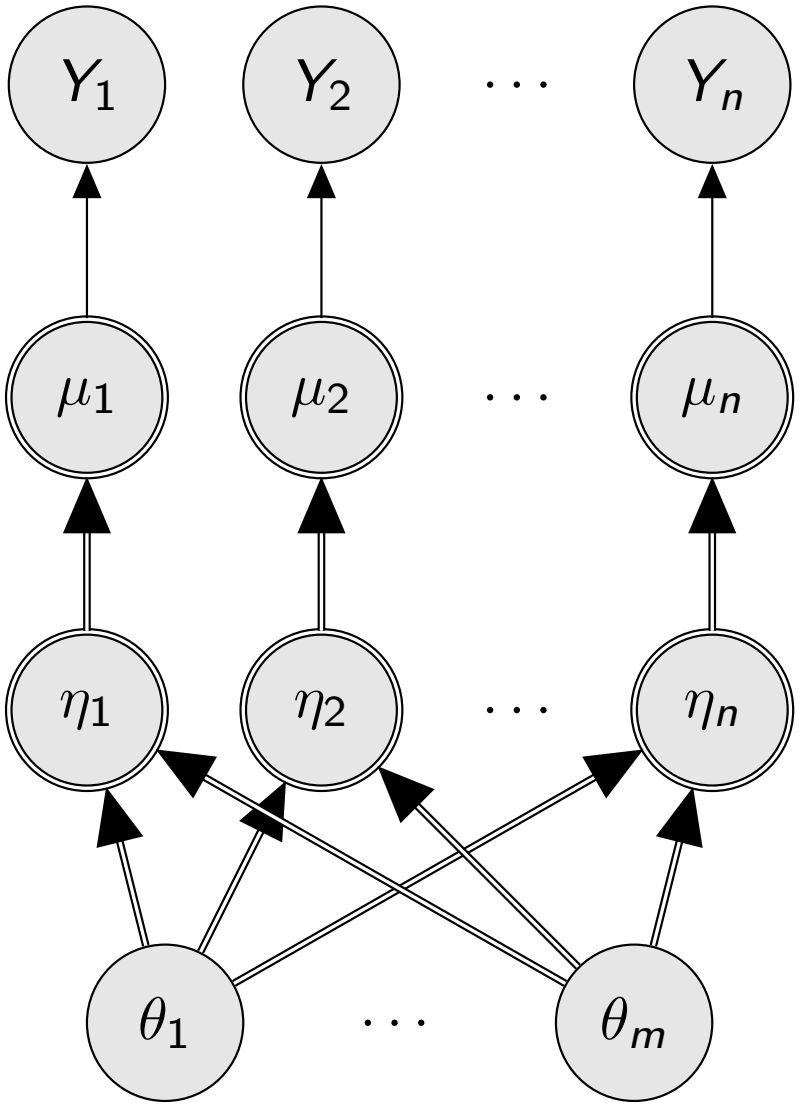
# Design Patterns




- Design patterns are reusable solutions to commonly recurring design problems
- Originally developed in architecture, the patterns concept has been translated to software development.
- This may be a useful way of thinking about efficient sampling of graphical models, which often have a rich structure.
- First we need to look for recurring design motifs

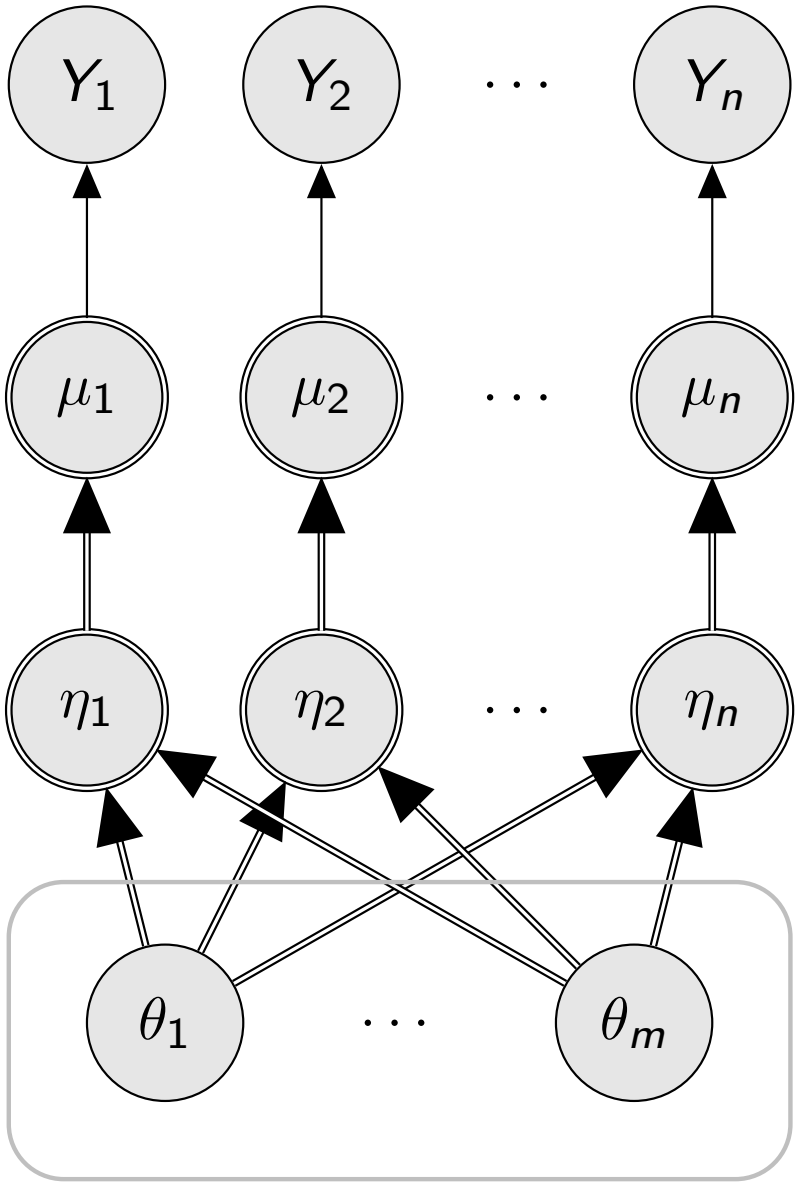
# GLM as a design motif




A GLM is a sub-graph with the following elements

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

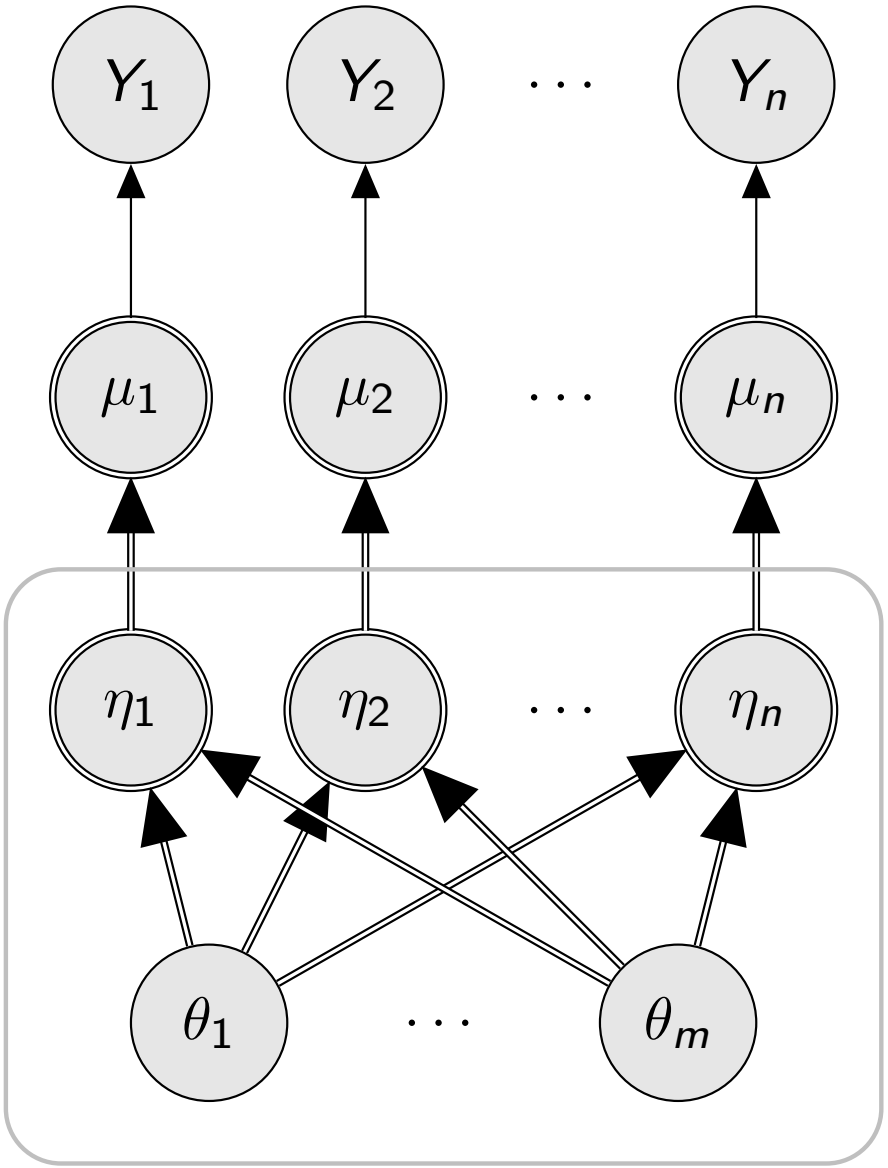
# GLM as a design motif




A GLM is a sub-graph with the following elements

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

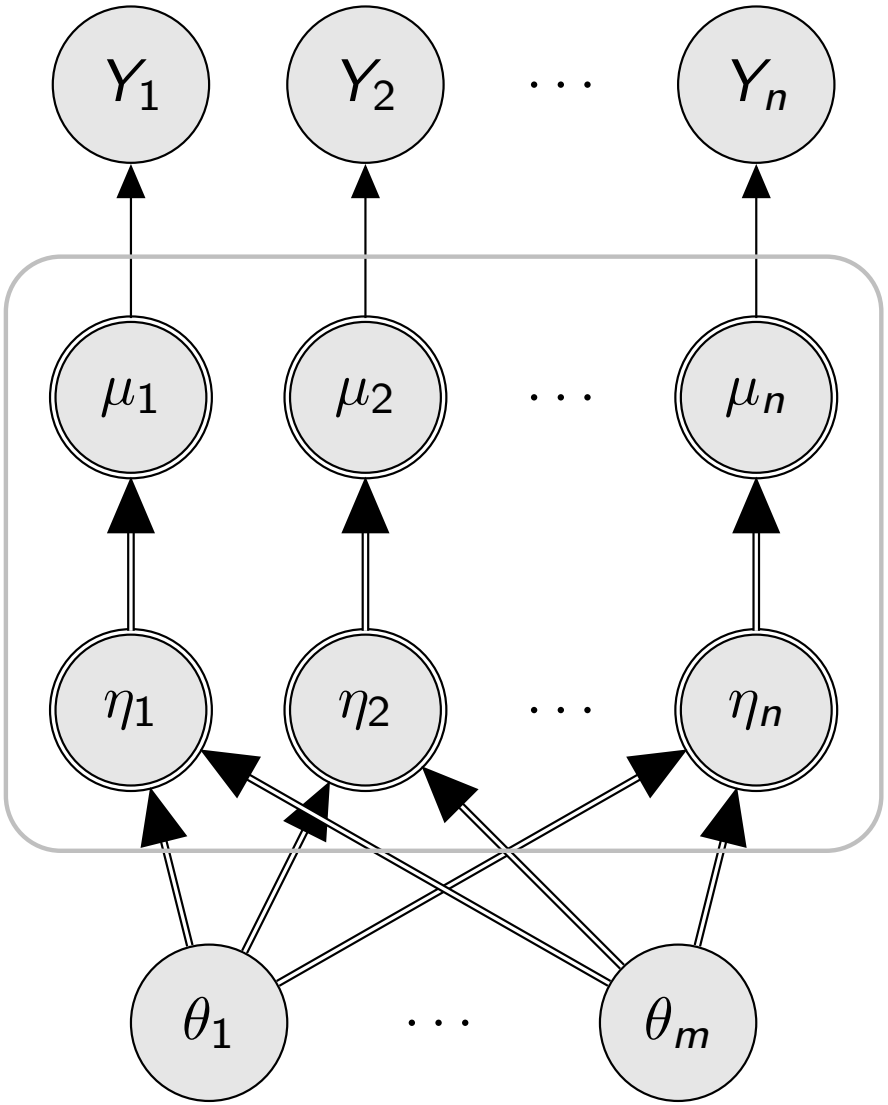
# GLM as a design motif




A GLM is a sub-graph with the following elements

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

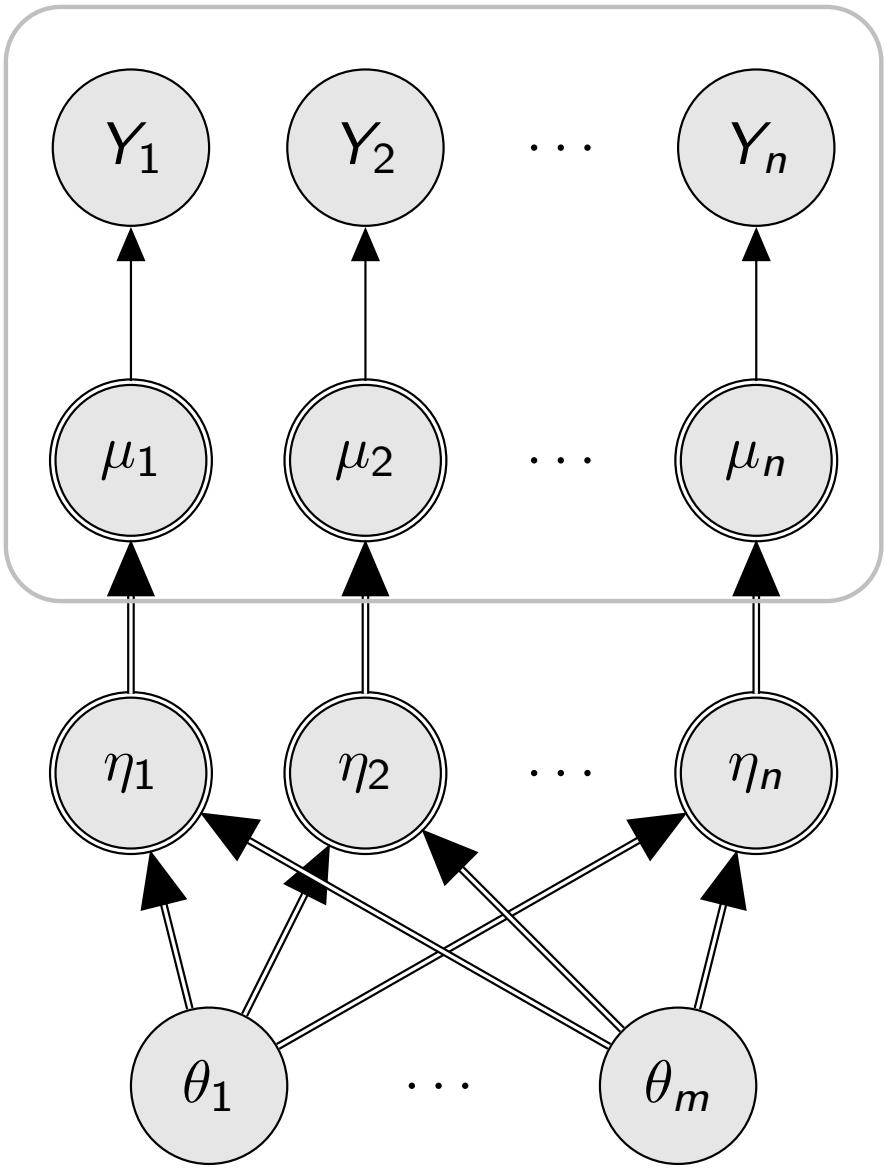
# GLM as a design motif




A GLM is a sub-graph with the following elements

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

# GLM as a design motif




A GLM is a sub-graph with the following elements

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

# Mixed models

- Mixed models are usually described using notation due to Laird and Ware (1982)

$$\begin{aligned} \boldsymbol{\eta} &= \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} \\ \mathbf{b} &\sim N(0, \Psi) \end{aligned}$$

- In software the distinction between fixed effects ($\boldsymbol{\beta}$) and random effects ($\mathbf{b}$) is usually implicit in the model syntax.
- If JAGS finds a GLM motif in the model, can it partition the nodes $\theta_1 \ldots \theta_m$ into fixed and random effects?

# Distinguishing between fixed and random effects

| Classical | Bayesian |
|---|---|
| Maximum likelihood | MCMC |
| $\boldsymbol{\beta}$ unknown parameter<br>$\mathbf{b}$ random variable | $\boldsymbol{\beta}, \mathbf{b}$ both random variables |
| Marginal likelihood<br>$p(\mathbf{Y} \mid \boldsymbol{\beta}, \Psi)$ | Local likelihood<br>$p(\mathbf{Y} \mid \boldsymbol{\beta}, \mathbf{b})$ |
| Need to estimate $\Psi$ | $\Psi$ fixed when updating $\boldsymbol{\beta}, \mathbf{b}$ |
| $\mathbf{X}$ dense<br>$\mathbf{Z}$ sparse | $(\mathbf{Z}, \mathbf{X})$ form a single design matrix with both dense and sparse components |

There are no mixed models, only models with sparse design matrices.

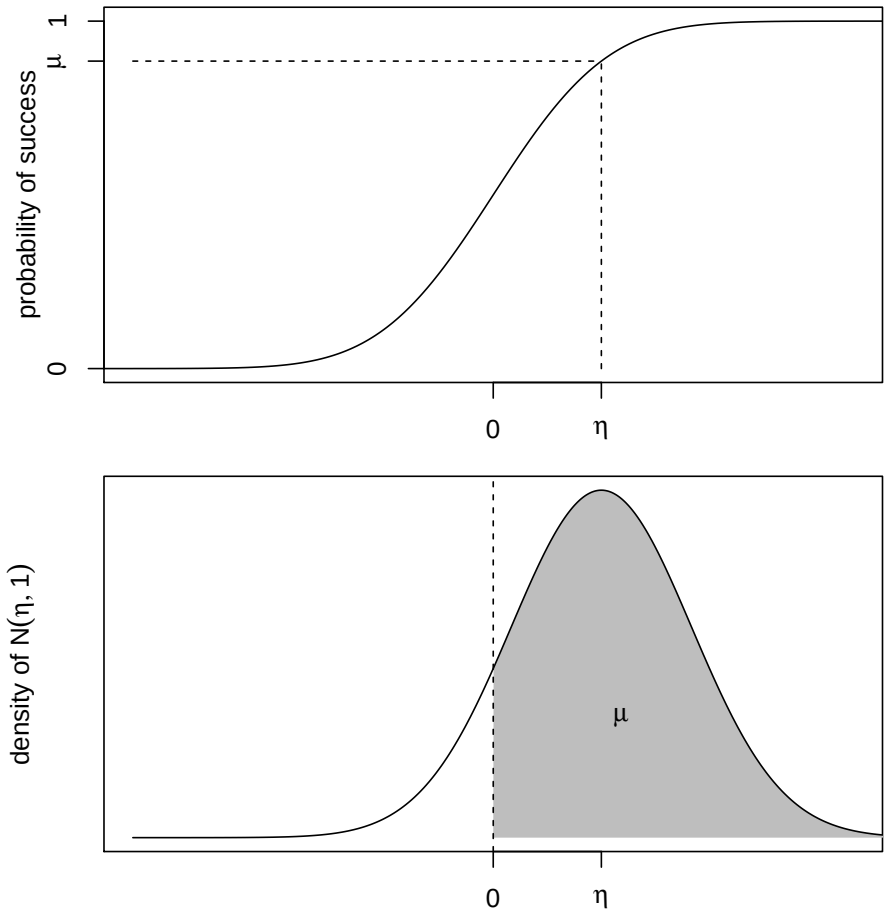
# Sparse matrix algebra in JAGS

- JAGS uses the sparse matrix libraries CSparse and CHOLMOD developed by Timothy A Davis.
- The same sampling engine handles both fixed-effect (dense) and mixed (sparse) linear models.
- The set of parameters of the linear model $\boldsymbol{\theta} = (\boldsymbol{\beta}, \mathbf{b})$ has multivariate normal posterior distribution, and so can be efficiently sampled.
- Programming is relatively easy due to the fundamental connection between graph theory and sparse matrix algebra

# Extending the linear model sampler

- 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

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

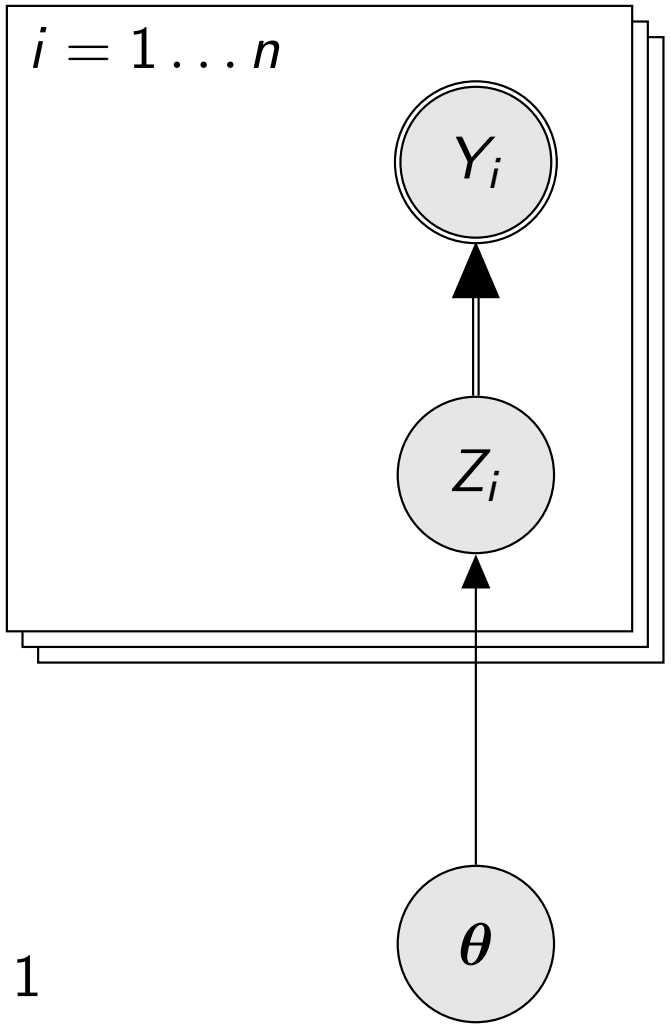
and make the outcome $Y$ a deterministic function of $Z$

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

# Graphical representation of Albert-Chib (1993)

Nodes $Z_1 \ldots Z_n$ are intermediate between the parameters $\theta$ and the outcomes $Y_1 \ldots Y_n$ so that

$$\theta \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$$

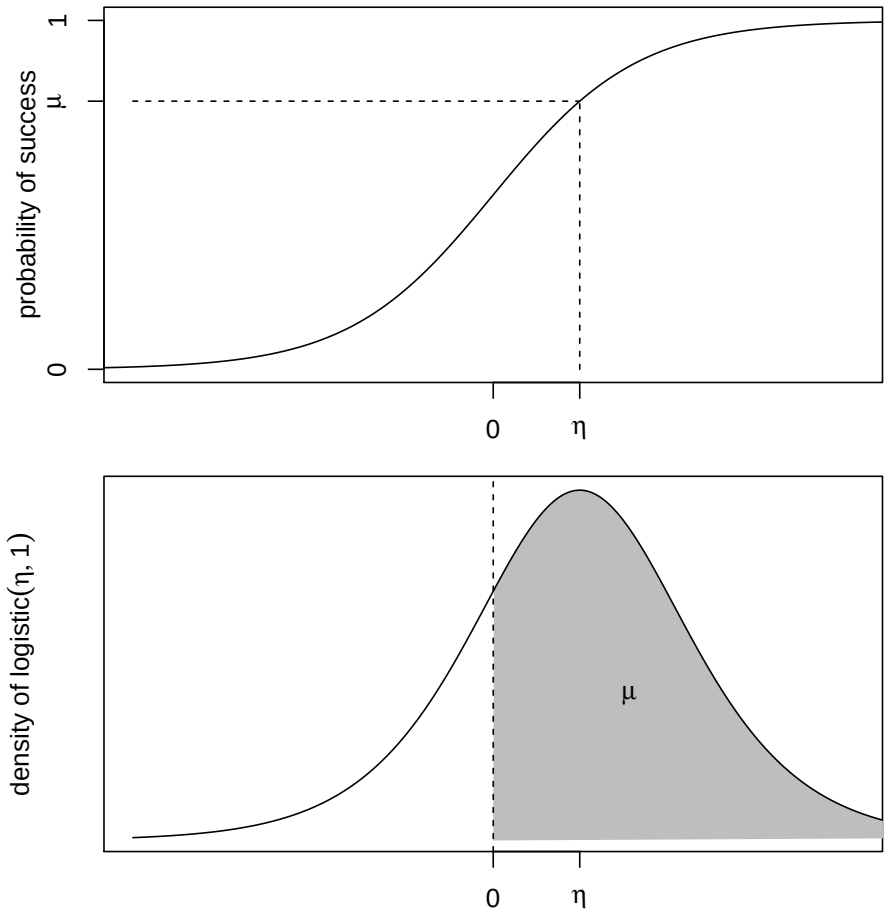



When sampling

- $\theta$ is updated given $\mathbf{Z}$ ignoring $\mathbf{Y}$, thereby reducing the problem to a linear model.
- $Z_1 \ldots Z_n$ are updated individually from the truncated normal

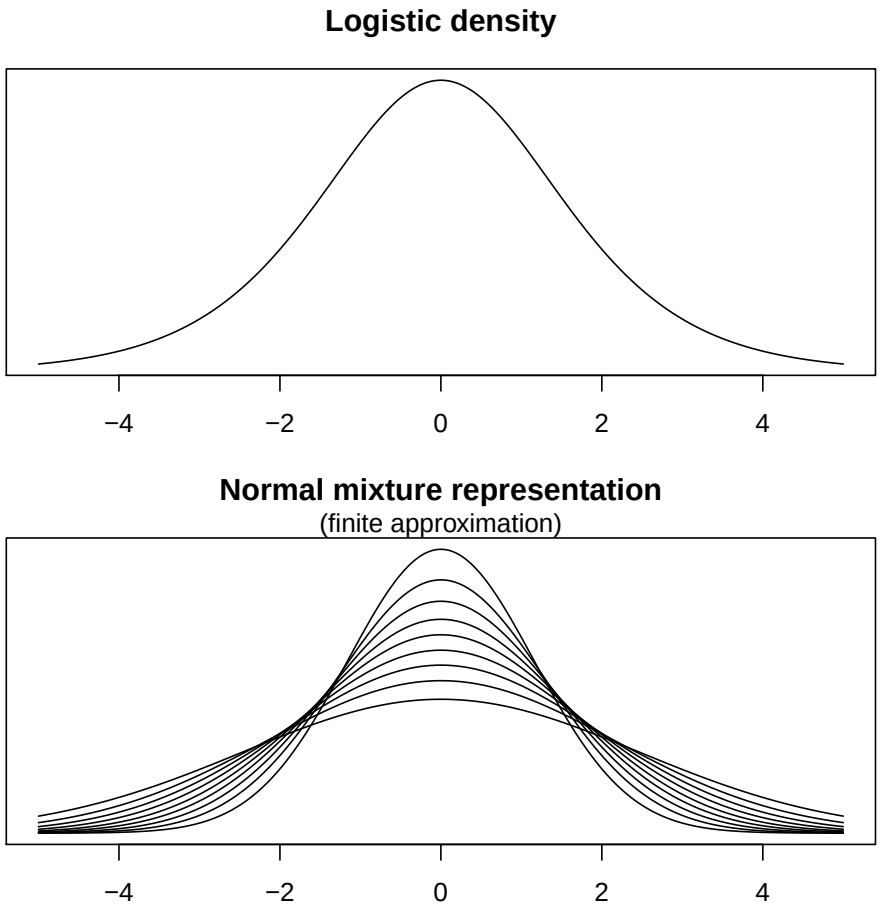
$$\begin{aligned} Z_i &\sim N(\eta_i, 1) I\{Z_i \geq 0\} \quad \text{if} \quad Y_i = 1 \\ Z_i &\sim N(\eta_i, 1) I\{Z_i < 0\} \quad \text{if} \quad Y_i = 0 \end{aligned}$$

# 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




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

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

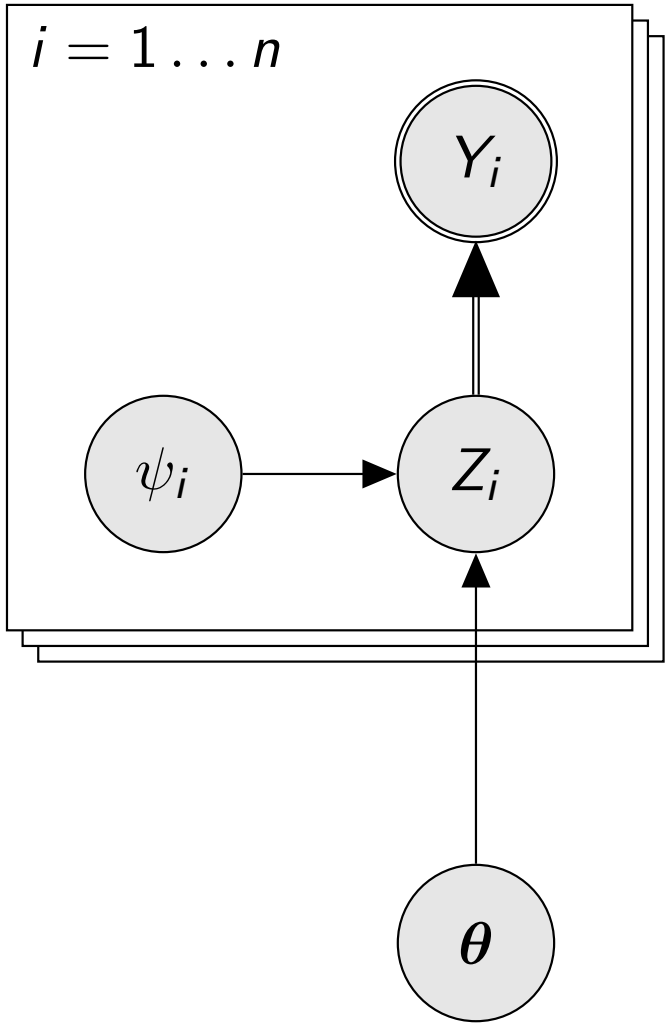
# Graphical representation of Holmes and Held logistic model

The logistic-mixture model adds further auxiliary nodes $\psi_1 \dots \psi_n$, such that

$$\psi_i \perp\!\!\!\perp \mathbf{Y} \mid Z_i$$

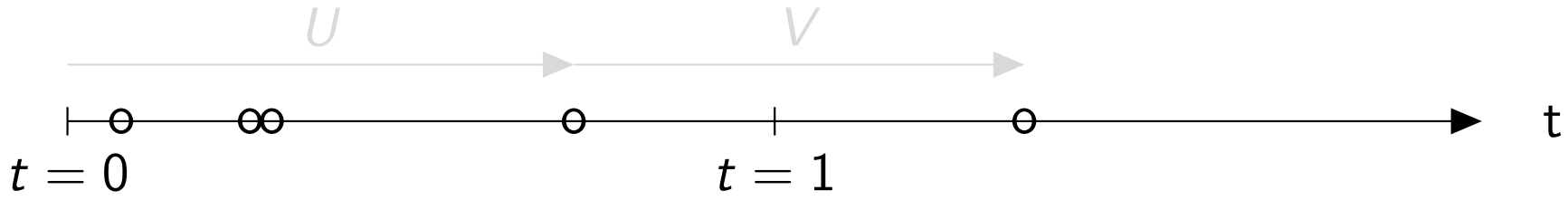
Hence $\psi_i$ is updated from $[\psi_i \mid Z_i, \boldsymbol{\theta}]$

- The Kolmogorov-Smirnov density has no closed-form expression.
- But Devroye (1986) provides two alternating series expansions that can be used for rejection sampling of $\psi_i$.
- $Z_i$ is updated from the truncated logistic distribution $Z_i \mid Y_i, \boldsymbol{\theta}$



# Frühwirth-Schnatter et al (2010) Poisson regression

In a Poisson regression model, $Y \sim \text{Po}(\lambda)$ where $\lambda = \exp(\eta)$. Frühwirth-Schnatter et al (2010) model $Y$ in terms of an underlying Poisson process of rate $\lambda$, as the number of events before time 1.

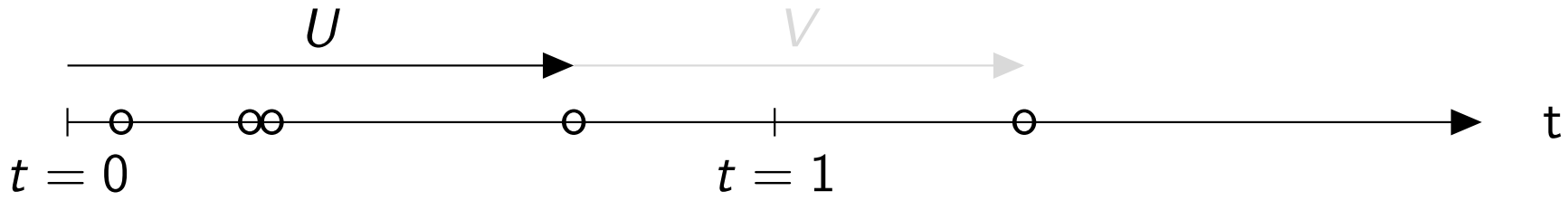



- Sufficient statistics for $\eta$ are
  - $U$, the last arrival before $t = 1$
  - $V$, the next inter-arrival time.
- On a log scale, the model is linear

$$\begin{aligned} -\log(U) &= \eta - \log(\epsilon) \quad \text{where} \quad \epsilon \sim \Gamma(y-1,1) \\ -\log(V) &= \eta - log(\xi) \quad \text{where} \quad \xi \sim \exp(1) \end{aligned}$$

# Frühwirth-Schnatter et al (2010) Poisson regression

In a Poisson regression model, $Y \sim \text{Po}(\lambda)$ where $\lambda = \exp(\eta)$. Frühwirth-Schnatter et al (2010) model $Y$ in terms of an underlying Poisson process of rate $\lambda$, as the number of events before time 1.

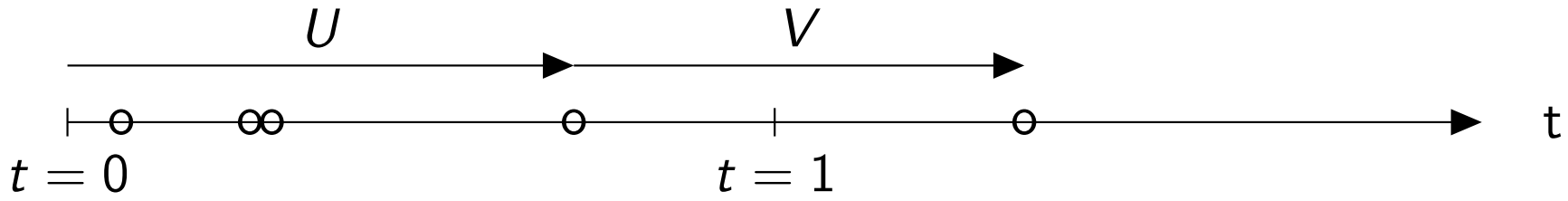



- Sufficient statistics for $\eta$ are
  - $U$, the last arrival before $t = 1$
  - $V$, the next inter-arrival time.
- On a log scale, the model is linear

$$\begin{aligned} -\log(U) &= \eta - \log(\epsilon) \quad \text{where} \quad \epsilon \sim \Gamma(y-1,1) \\ -\log(V) &= \eta - log(\xi) \quad \text{where} \quad \xi \sim \exp(1) \end{aligned}$$

# Frühwirth-Schnatter et al (2010) Poisson regression

In a Poisson regression model, $Y \sim \text{Po}(\lambda)$ where $\lambda = \exp(\eta)$. Frühwirth-Schnatter et al (2010) model $Y$ in terms of an underlying Poisson process of rate $\lambda$, as the number of events before time 1.

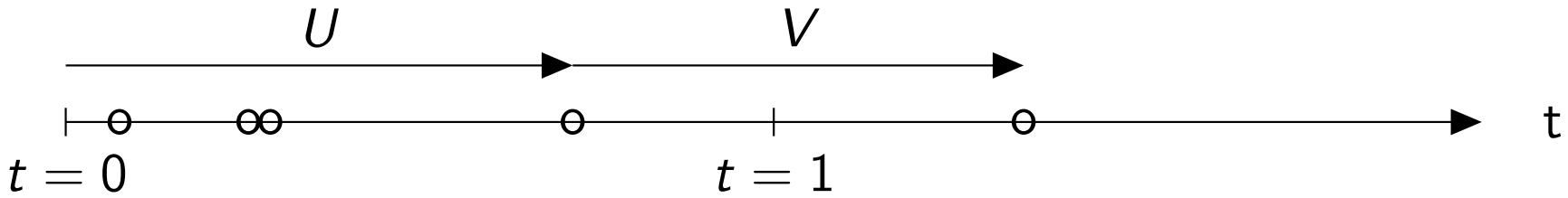



- Sufficient statistics for $\eta$ are
  - $U$, the last arrival before $t = 1$
  - $V$, the next inter-arrival time.
- On a log scale, the model is linear

$$\begin{aligned} -\log(U) &= \eta - \log(\epsilon) \quad \text{where} \quad \epsilon \sim \Gamma(y-1,1) \\ -\log(V) &= \eta - log(\xi) \quad \text{where} \quad \xi \sim \exp(1) \end{aligned}$$

# Frühwirth-Schnatter et al (2010) Poisson regression

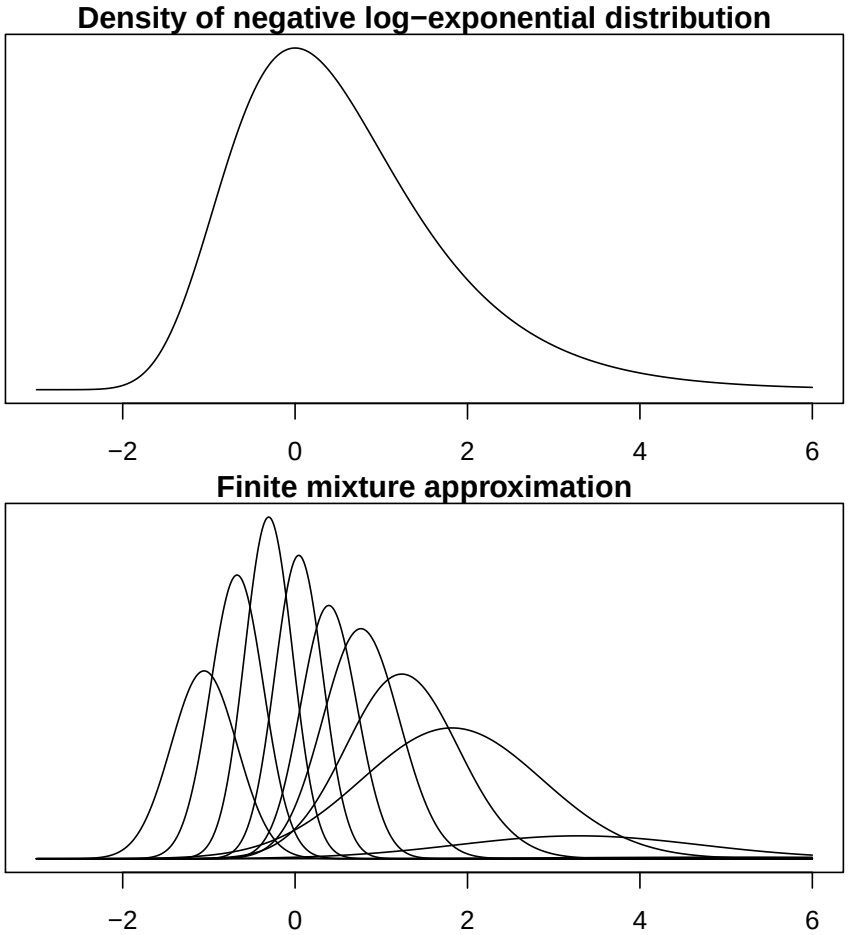
In a Poisson regression model, $Y \sim \text{Po}(\lambda)$ where $\lambda = \exp(\eta)$. Frühwirth-Schnatter et al (2010) model $Y$ in terms of an underlying Poisson process of rate $\lambda$, as the number of events before time 1.




- Sufficient statistics for $\eta$ are
  - $U$, the last arrival before $t = 1$
  - $V$, the next inter-arrival time.
- On a log scale, the model is linear

$$\begin{aligned} -\log(U) &= \eta - \log(\epsilon) \quad \text{where} \quad \epsilon \sim \Gamma(y-1,1) \\ -\log(V) &= \eta - log(\xi) \quad \text{where} \quad \xi \sim \exp(1) \end{aligned}$$

# Mixture representation of negative log-gamma




The GMRFlib library contains code for approximating the negative log-gamma distribution as a finite mixture of normals. This code is borrowed by JAGS, under the GPL.

Using this mixture approximation reduced the model to a normal linear model.

# Frühwirth-Schnatter et al (2010) logistic regression

In a logistic regression model $Y \sim \text{Bin}(\pi, n)$ where $\pi = \lambda/(1+\lambda)$ and $\lambda = \exp(\eta)$. Frühwirth-Schnatter et al (2010) model $Y$ as the sum of $n$ Bernoulli trials with probability $\pi$.

$$Y = \sum_{i=1}^{n} Y_j$$

In each trial, $Y_j$ is a function of two latent variables $U_j, V_j$

$$\begin{aligned} U_j &\sim \exp(\lambda) \\ V_j &\sim \exp(1) \\ Y_j &= I\{U_j < V_j\} \end{aligned}$$

The sufficient statistic for $\eta$ is $U = \sum_j U_j$ and

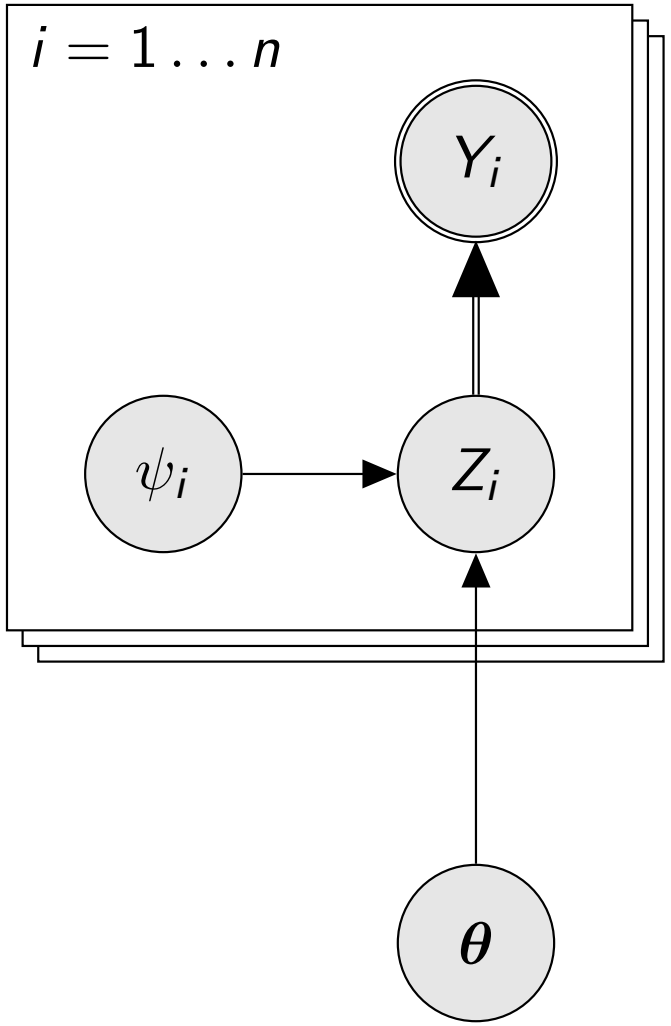
$$-\log(U) = \eta - \log(\epsilon) \text{ where } \epsilon \sim \Gamma(n, 1)$$

The same mixture representation for $-\log(\epsilon)$ reduces the model to a normal linear model

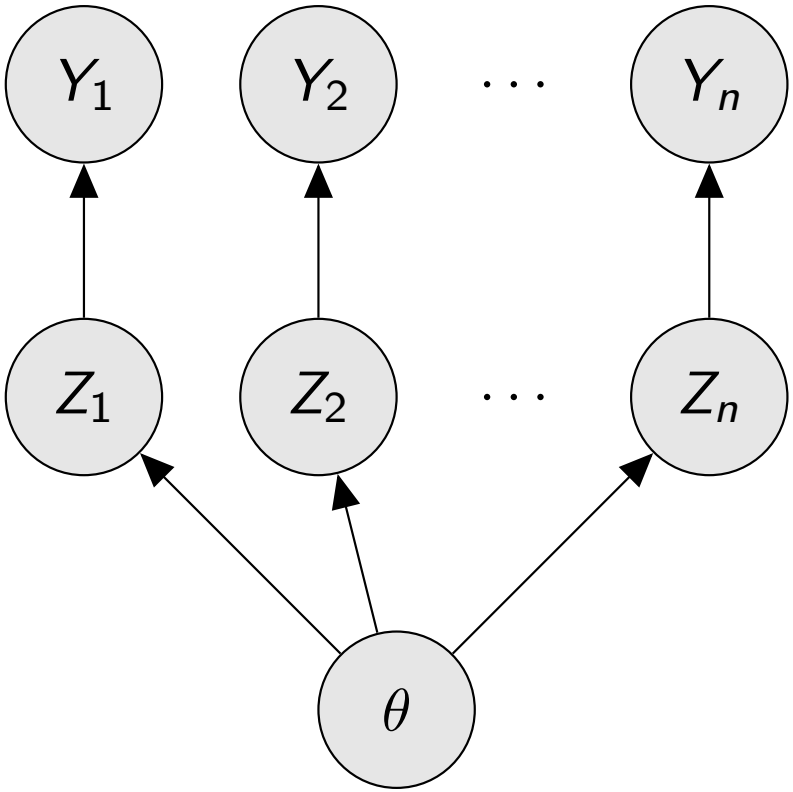
# Graphical representation

These Poisson and logistic regression models have the same graphical representation as the Holmes and Held logistic model.

- $\psi_i$ is an integer value that determines which mixture component is in use.
- $Z_i = (U_i, V_i)$ can be efficiently sampled given $\boldsymbol{\theta}$, $Y_i$ and marginalizing over $\psi_i$.
- Sampling of $\psi_i$ given $Z_i, \boldsymbol{\theta}$ is trivial.



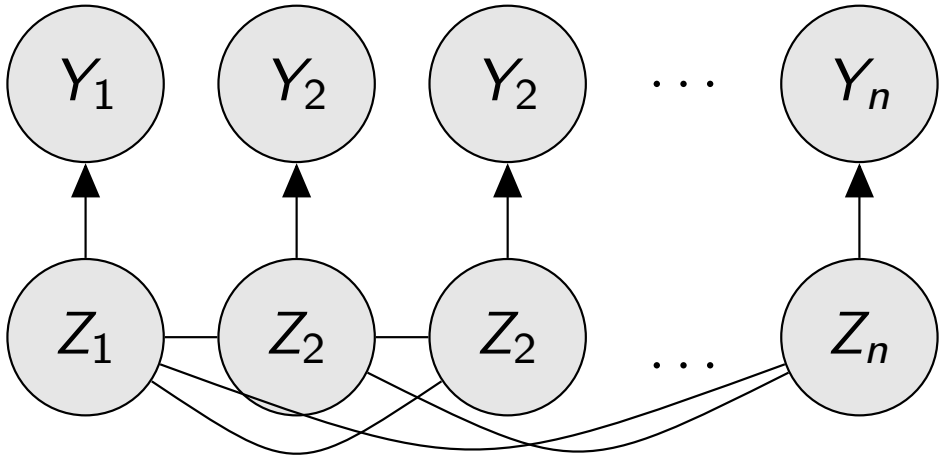
# Patterns again




- All of these graphical contain the same distinctive "fantail" motif.
- Whenever this motif is seen in a graph, Gibbs sampling (*i.e.* sampling individual nodes) produces poor mixing.
- When $\mathbf{Z}$ is sampled, it depends strongly on the current value of $\theta$
- Conversely, when $\theta$ is sampled, it depends strongly on the current value of $\mathbf{Z}$.

# Improved binary probit sampler

Holmes and Held (2006) suggested an improved sampler for the probit model that removes the dependency of $Z$ on $\boldsymbol{\theta}$. When updating $Z$, we integrate out $\boldsymbol{\theta}$.




In this marginal model, $\mathbf{Z}$ has a joint multivariate normal prior.

- $\mathbf{Z}$ is updated element-wise
- $[Z_i \mid Y_i, \mathbf{Z}_{-i}]$ has a truncated normal distribution. Its mean and variance can be calculated from the posterior variance of $\boldsymbol{\theta} \mid Z$, which is already calculated by the update method for $\boldsymbol{\theta}$.
- This produces substantial increase in efficiency of the sampler

# Improved binary-logit sampler?

Holmes and Held (2006) extended the marginal sampling method to the binary logistic model, but found little improvement

- To preserve multivariate normality of $\mathbf{Z}$, we need to condition on the current values of the mixture parameters $\boldsymbol{\psi}$
- But the coverage of the normal mixture component $Z_i \mid \psi_i$ can be much smaller than the full distribution of $Z_i$.
- This reduces the step-size for the update of each element $Z_i$ and so reduces mixing.

# Work in progress...

The marginal sampler can be generalized to all auxiliary variable models

- Break down sampling of $[Z_i \mid \mathbf{Z}_{-i}, Y_i]$ into two steps
  1. Sample $[\eta^{(i)} \mid \mathbf{Z}_{-i}, \boldsymbol{\psi}_{-i}, Y_i]$
  2. Sample $[Z_i \mid \eta^{(i)}, Y_i]$

  where $\eta^{(i)}$ is a private version of the linear predictor $\eta_i = \mathbf{x}_i^T \boldsymbol{\theta}$.
- This is an example of redundant parameterization (breaking a single identifiable node into separate non-identifiable components)
- Step 2 is the same as before
- Step 1 is trivial for binary-logit and Poisson models...

# Conclusions

- Data augmentation (adding nodes) simplifies update methods for GLMs but may lead to poor sampling performance.
- Marginalization (removing nodes) removes unwanted dependencies between nodes, improving sampling performance.
- Redundant parameterization (splitting an identifiable node into non-identifiable components) combines features of both.