WinBUGS Tutorial Outline

Ed Merkle & Trisha Van Zandt

August 4, 2005

- WinBUGS overview
- Description of Example Data
- Example 1: Confidence in Signal Detection
- Example 2: Mean RT Model
- Example 3: Estimating RT Distributions
- Conclusions/Things to Remember

WinBUGS Overview

- **WinBUGS: Making MCMC available to applied researchers.**

WinBUGS Overview

- **WinBUGS: Making MCMC available to applied researchers.**
- **Can download over the internet; obtain free key for unrestricted use.**

WinBUGS Overview

- **WinBUGS: Making MCMC available to applied researchers.**
- **Can download over the internet; obtain free key for unrestricted use.**
- **Similar programs available for Linux users; also for R.**

WinBUGS Overview

- **WinBUGS: Making MCMC available to applied researchers.**
- **Can download over the internet; obtain free key for unrestricted use.**
- **Similar programs available for Linux users; also for R.**
- **Workshop will focus exclusively on WinBUGS.**

What Does WinBUGS Do?

- **Sample parameters from the appropriate posterior distribution; summarize large samples.**

What Does WinBUGS Do?

- **Sample parameters from the appropriate posterior distribution; summarize large samples.**

- **Depending on situation, different sampling methods are employed:**

What Does WinBUGS Do?

- **Sample parameters from the appropriate posterior distribution; summarize large samples.**

- **Depending on situation, different sampling methods are employed:**
  - **Conjugate priors: Direct sampling**
  - **Log-concave distributions: Adaptive rejection sampling**
  - **Non log-concave: Slice sampling, Metropolis-Hastings**

What Does WinBUGS Do?

- **Sample parameters from the appropriate posterior distribution; summarize large samples.**

- **Depending on situation, different sampling methods are employed:**
  - **Conjugate priors: Direct sampling**
  - **Log-concave distributions: Adaptive rejection sampling**
  - **Non log-concave: Slice sampling, Metropolis-Hastings**

- **Aside from direct sampling, other methods generally work by sampling from distributions similar to the posterior, then accepting the sample with some probability.**

General Program Layout

- **Three main files/specifications needed:**
  - **Program file with model specification/priors**
  - **Data file in a specific format**
  - **File of parameter starting values**

General Program Layout

- **Three main files/specifications needed:**
  - **Program file with model specification/priors**
  - **Data file in a specific format**
  - **File of parameter starting values**
- **Starting values are optional, as WinBUGS can generate starting values.**
  - **The generated values can cause the program to crash, though!**

General Program Layout

- **Three main files/specifications needed:**
  - **Program file with model specification/priors**
  - **Data file in a specific format**
  - **File of parameter starting values**
- **Starting values are optional, as WinBUGS can generate starting values.**
  - **The generated values can cause the program to crash, though!**
- **Similarity to R/Splus**

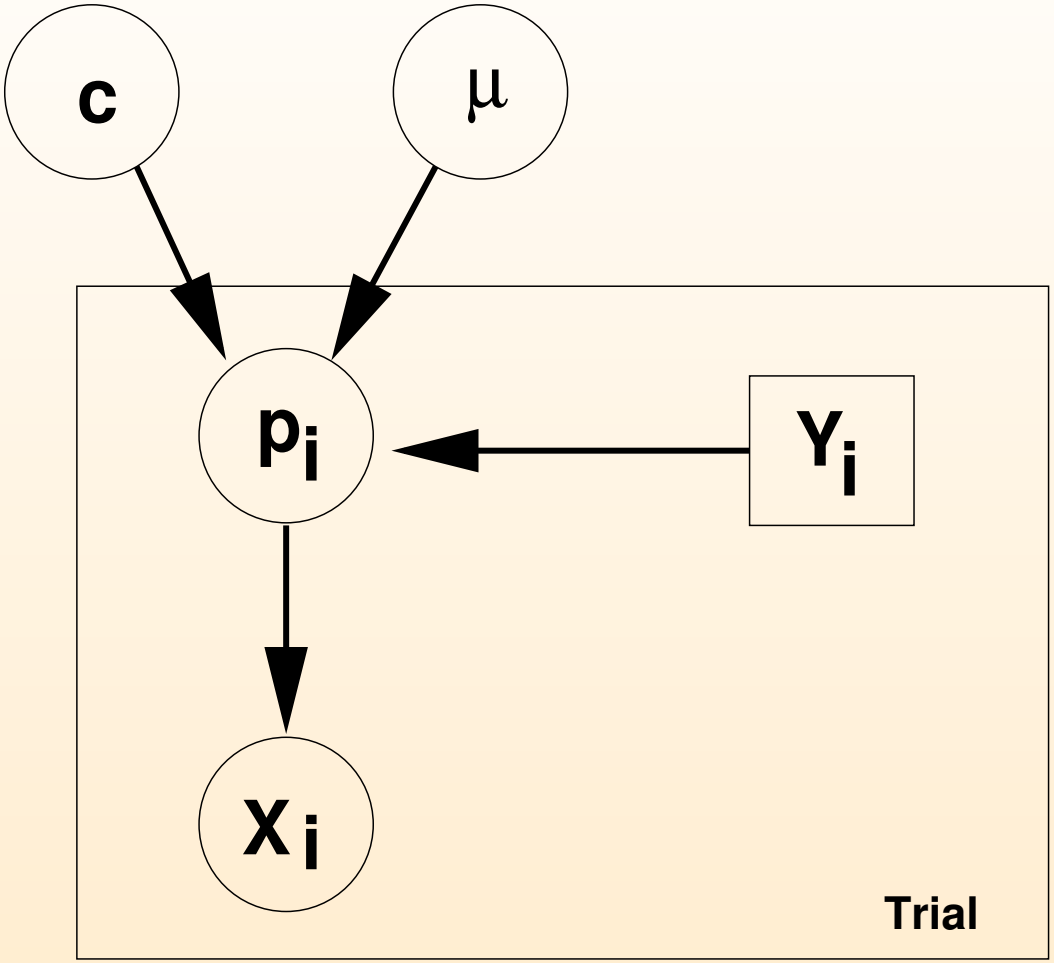
Model Specification

- **Define data distributions, prior distributions**
  - **Distributions defined using ~ operator**
  - **Assignment defined using the <- operator**

Short SDT example:

```
model
{
# x is vector of observed responses (1/2)
# y is vector of correct response

# Define x as categorical with probability of response
# determined by SDT model.
for (i in 1:N){
  x[i] ~ dcat(p[i,])

  # Define SDT probabilities, assuming variance of 0.5
  p[i,1] <- phi((c - mu*y[i])/0.5)
  p[i,2] <- 1 - p[i,1]
}

# Define prior distributions on model parameters.
mu ~ dnorm(0,0.0001)
c ~ dnorm(0,0.0001)
}

list(mu=0.5,c=0.25)
```

SDT Doodle



Data Specification/Starting Values

- **Two accepted data formats:**
  - **R/Splus format: Use the "dput" command in R to put formatted data in a text file.**
  - **Rectangular format: Text file containing variables in separate columns.**

Data Specification/Starting Values

- **Two accepted data formats:**
  - **R/Splus format: Use the "dput" command in R to put formatted data in a text file.**
  - **Rectangular format: Text file containing variables in separate columns.**

- **If using R format, matrices must be entered with care:**
  - **R reads/writes down columns, while WinBUGS reads across rows.**
  - **Must manually add a ".Data=" command before the matrix entries.**

Data Specification/Starting Values

- **Two accepted data formats:**
  - **R/Splus format: Use the "dput" command in R to put formatted data in a text file.**
  - **Rectangular format: Text file containing variables in separate columns.**

- **If using R format, matrices must be entered with care:**
  - **R reads/writes down columns, while WinBUGS reads across rows.**
  - **Must manually add a ".Data=" command before the matrix entries.**

- **Starting value formats are just like data formats.**

Initializing the Model

- **Submit model specification & data to WinBUGS using Model Specification Tool:**
  - **Highlight "model", click "check model".**
  - **Highlight "list", click "load data".**
  - **Type number of chains, click "compile".**
  - **Highlight "list" of starting values, click "load inits".**
  - **If generating starting values, click "gen inits".**

Initializing the Model

- **Submit model specification & data to WinBUGS using Model Specification Tool:**
  - **Highlight "model", click "check model".**
  - **Highlight "list", click "load data".**
  - **Type number of chains, click "compile".**
  - **Highlight "list" of starting values, click "load inits".**
  - **If generating starting values, click "gen inits".**

- **Error messages appear in lower right-hand corner.**

Initializing the Model

- **Submit model specification & data to WinBUGS using Model Specification Tool:**
  - **Highlight "model", click "check model".**
  - **Highlight "list", click "load data".**
  - **Type number of chains, click "compile".**
  - **Highlight "list" of starting values, click "load inits".**
  - **If generating starting values, click "gen inits".**
- **Error messages appear in lower right-hand corner.**
- **When "model is initialized" message appears, you are ready to simulate.**

Running the Simulation

- **Click "Samples..." off the inference menu to bring up the Sample Monitor Tool.**
  - **Type individual parameter names, click "set"**
  - **If sampling method requires an adaptive phase, not all options will initially be available to you.**

Running the Simulation

- **Click "Samples..." off the inference menu to bring up the Sample Monitor Tool.**
  - **Type individual parameter names, click "set"**
  - **If sampling method requires an adaptive phase, not all options will initially be available to you.**
- **Click "Update..." off the model menu.**
  - **Type desired number of iterations & refresh frequency, then click update button to run the simulation.**
  - **If program crashes, changing the starting values can solve the problem.**

Assessing Convergence

- **Parameter chains do not immediately converge to the posterior distribution; multiple tools exist for assessing convergence in WinBUGS.**

Assessing Convergence

- **Parameter chains do not immediately converge to the posterior distribution; multiple tools exist for assessing convergence in WinBUGS.**
  - **Time series plots: Horizontal bands with no upward/downward trends imply convergence.**

Assessing Convergence

- **Parameter chains do not immediately converge to the posterior distribution; multiple tools exist for assessing convergence in WinBUGS.**
  - **Time series plots: Horizontal bands with no upward/downward trends imply convergence.**
  - **Autocorrelation plots: High autocorrelations imply chains that are slow to converge.**

Assessing Convergence

- **Parameter chains do not immediately converge to the posterior distribution; multiple tools exist for assessing convergence in WinBUGS.**
  - **Time series plots: Horizontal bands with no upward/downward trends imply convergence.**
  - **Autocorrelation plots: High autocorrelations imply chains that are slow to converge.**
  - **Gelman-Rubin plot: Horizontal lines; red line close to 1 implies convergence.**

Assessing Convergence

- **Parameter chains do not immediately converge to the posterior distribution; multiple tools exist for assessing convergence in WinBUGS.**
  - **Time series plots: Horizontal bands with no upward/downward trends imply convergence.**
  - **Autocorrelation plots: High autocorrelations imply chains that are slow to converge.**
  - **Gelman-Rubin plot: Horizontal lines; red line close to 1 implies convergence.**

- **Initial sampled parameters are usually discarded to allow for burn-in/convergence to posterior distribution.**

Summarizing Results

- **Sample monitor tool provides many useful options:**
  - **"stats" yields summary statistics.**
  - **"history", "auto cor" yield time series, autocorrelation plots.**
  - **"bgr diag" yields Gelman-Rubin statistic plots.**

Summarizing Results

- Sample monitor tool provides many useful options:
  - "stats" yields summary statistics.
  - "history", "auto cor" yield time series, autocorrelation plots.
  - "bgr diag" yields Gelman-Rubin statistic plots.

- Comparison Tool provides summary plots for parameter vectors, plots of observed vs. predicted data.

Summarizing Results

- **Sample monitor tool provides many useful options:**
  - **"stats" yields summary statistics.**
  - **"history", "auto cor" yield time series, autocorrelation plots.**
  - **"bgr diag" yields Gelman-Rubin statistic plots.**
- **Comparison Tool provides summary plots for parameter vectors, plots of observed vs. predicted data.**
- **DIC Tool provides DIC measure of model complexity.**

Example Data

- **Data from same experiment are used for all 3 examples.**

- **Participants presented with dots clusters, where number of dots arose from one of two normal distributions differing in mean number of dots.**

Example Data

- Data from same experiment are used for all 3 examples.
- Participants presented with dots clusters, where number of dots arose from one of two normal distributions differing in mean number of dots.
- For each cluster, participants state confidence that dots arose from the high distribution.

Example Data

- **Data from same experiment are used for all 3 examples.**
- **Participants presented with dots clusters, where number of dots arose from one of two normal distributions differing in mean number of dots.**
- **For each cluster, participants state confidence that dots arose from the high distribution.**
- **Collected data:**
  - **Confidence: From 5% to 95% in increments of 10%.**
  - **Correct Distribution: True distribution from which the dots arose.**
  - **Condition: Easy condition $\mu_H = 55$ and $\mu_L = 45$; Hard condition $\mu_H = 53$ and $\mu_L = 47$.**
  - **Implied Response: Implied choice based on the confidence judgments.**
  - **Number of Dots: Number of dots in a specific dots cluster.**
  - **RT: The participant's confidence response time in msec.**

Disclaimer

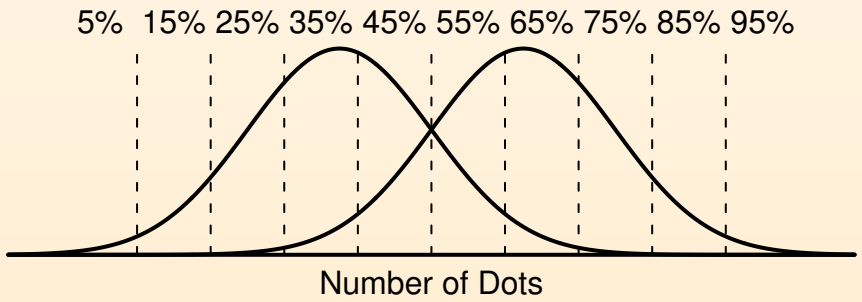
In the examples that follow, we estimate relatively simple models using relatively simple data sets. The models are intended to introduce WinBUGS to a general math psych audience; they are not intended to be optimal for the given data sets. Furthermore, we do not recommend blind application of these models to future data sets.

Example 1: Confidence in Signal Detection

- **Signal detection model of confidence:**
  - **Partitions placed on signal/noise distributions**
  - **Confidence determined by which two partitions a given stimulus falls between.**
  - **Similar to Ferrell & McGoey's (1980) Decision Variable Partition Model.**

Example 1: Confidence in Signal Detection

- **Signal detection model of confidence:**
  - **Partitions placed on signal/noise distributions**
  - **Confidence determined by which two partitions a given stimulus falls between.**
  - **Similar to Ferrell & McGoey's (1980) Decision Variable Partition Model.**



The Model

- **Mathematically:**
  - x **is vector of confidence judgments**
  - γ **is vector of partitions**
  - K **is distribution from which dots actually arose**
  - **Mean (**μ_k**) of each distribution and common standard deviation (**s**) are assumed known.**
  - $\gamma_1 = -\infty$**;** $\gamma_{11} = \infty$**.**

The Model

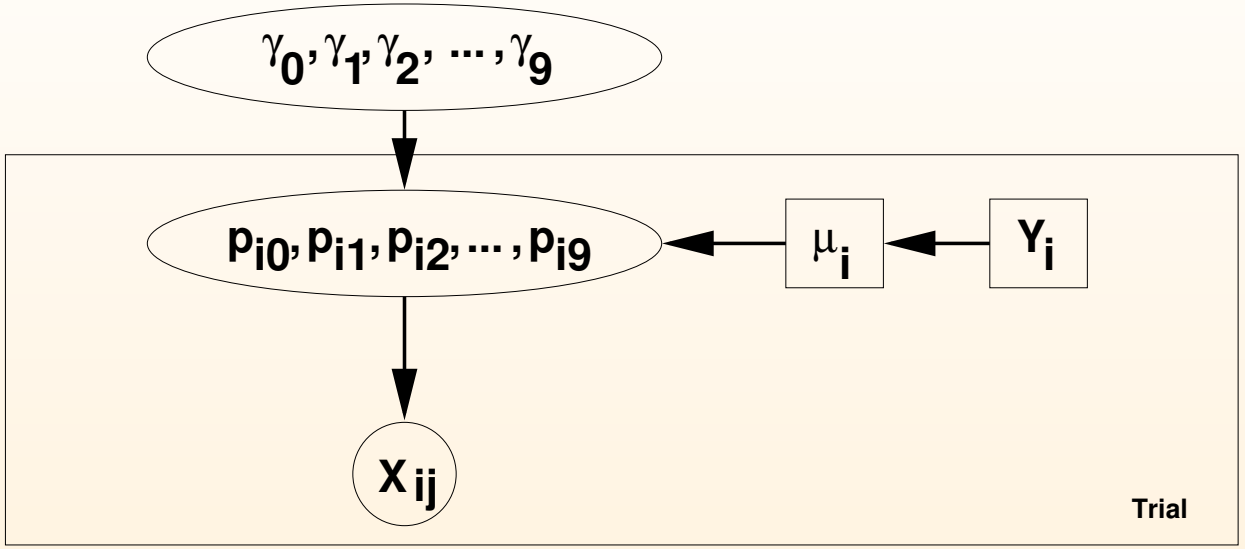
- **Mathematically:**
  - x **is vector of confidence judgments**
  - γ **is vector of partitions**
  - K **is distribution from which dots actually arose**
  - **Mean (**μ_k**) of each distribution and common standard deviation (**s**) are assumed known.**
  - $\gamma_1 = -\infty$**;** $\gamma_{11} = \infty$**.**

- $P(x_i = j \mid K = k) = \Phi((\gamma_j - \mu_k)/s) - \Phi((\gamma_{j-1} - \mu_k)/s)$**.**

Model Specification

```
model
{
# y is vector of distribution from which dots arose
# x is vector of raw confidence data
mu[1] <- 45
mu[2] <- 55

for (i in 1:2){
  p[i,1] <- phi((gamm[1] - mu[i])/5)
  for (j in 2:9){
    p[i,j] <- phi((gamm[j]-mu[i])/5) - phi((gamm[j-1]-mu[i])/5)
  }
  p[i,10] <- 1 - phi((gamm[9] - mu[i])/5)
}

for (k in 1:N){
  x[k] ~ dcat(p[y[k]+1,])
}

gamm[1] ~ dnorm(50, 1.0E-06)I(,gamm[2])
for (j in 2:8){
  gamm[j] ~ dnorm(50, 1.0E-06)I(gamm[j-1],gamm[j+1])
}
gamm[9] ~ dnorm(50, 1.0E-06)I(gamm[8],)
}
```

Model Doodle



Notes on the Model Specification

- **Confidence judgments are modeled as categorical variables, with probabilities based on the threshold parameters and the distribution from which the dots arose.**

Notes on the Model Specification

- **Confidence judgments are modeled as categorical variables, with probabilities based on the threshold parameters and the distribution from which the dots arose.**

- **The γ vector is of length 9 instead of 11, as it was when the model was initially described. This is because WinBUGS did not like the 11×1 vector γ containing the fixed parameters $\gamma_1 = -\infty$ and $\gamma_{11} = +\infty$.**

Notes on the Model Specification

- **Confidence judgments are modeled as categorical variables, with probabilities based on the threshold parameters and the distribution from which the dots arose.**

- **The γ vector is of length 9 instead of 11, as it was when the model was initially described. This is because WinBUGS did not like the 11×1 vector γ containing the fixed parameters $\gamma_1 = -\infty$ and $\gamma_{11} = +\infty$.**

- **Ordering of the parameters in γ is controlled by the prior distributions. Each parameter's prior distribution is essentially flat and is censored by the other threshold parameters surrounding it using the "I(,)" specification. For another example on the use of ordered thresolds, see the "Inhalers" example included with WinBUGS.**

Notes on the Model Specification

- **Confidence judgments are modeled as categorical variables, with probabilities based on the threshold parameters and the distribution from which the dots arose.**

- **The γ vector is of length 9 instead of 11, as it was when the model was initially described. This is because WinBUGS did not like the 11×1 vector γ containing the fixed parameters $\gamma_1 = -\infty$ and $\gamma_{11} = +\infty$.**

- **Ordering of the parameters in γ is controlled by the prior distributions. Each parameter's prior distribution is essentially flat and is censored by the other threshold parameters surrounding it using the "I(,)" specification. For another example on the use of ordered thresolds, see the "Inhalers" example included with WinBUGS.**

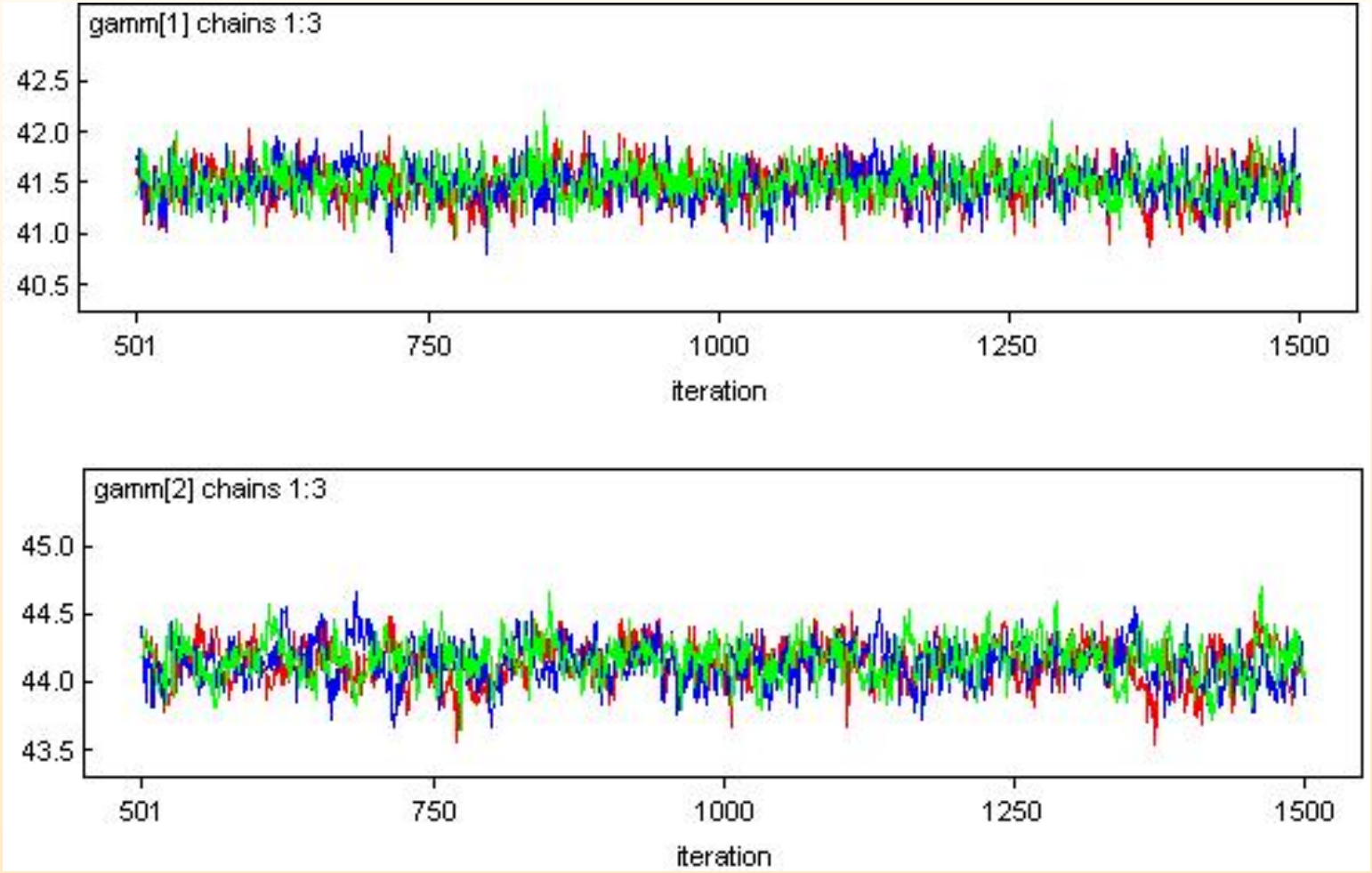
- **Sampling from this model is slow; it takes a couple minutes per 1000 iterations. It is thought that the censoring causes this delay.**

Results

- **3 parameter chains sampled for 1500 iterations each; first 500 discarded as burn-in.**

Results

- **3 parameter chains sampled for 1500 iterations each; first 500 discarded as burn-in.**



Results

- **Other tools give further evidence of convergence.**

Results

- **Other tools give further evidence of convergence.**
- **Summary statistics for threshold parameters:**

Node statistics

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
gamm[1]	41.49	0.1859	0.00568	41.11	41.49	41.84	501	3000
gamm[2]	44.14	0.1526	0.006595	43.84	44.15	44.43	501	3000
gamm[3]	46.3	0.1393	0.007265	46.03	46.31	46.58	501	3000
gamm[4]	47.71	0.1348	0.007639	47.45	47.71	47.98	501	3000
gamm[5]	49.02	0.1285	0.007446	48.77	49.01	49.27	501	3000
gamm[6]	50.45	0.124	0.006738	50.2	50.44	50.69	501	3000
gamm[7]	52.19	0.127	0.006412	51.95	52.19	52.44	501	3000
gamm[8]	54.38	0.1345	0.005889	54.11	54.38	54.65	501	3000
gamm[9]	57.09	0.1566	0.005353	56.77	57.09	57.39	501	3000

Example 2: A Model for Mean RT

- **The relationship between RT and difficulty.**

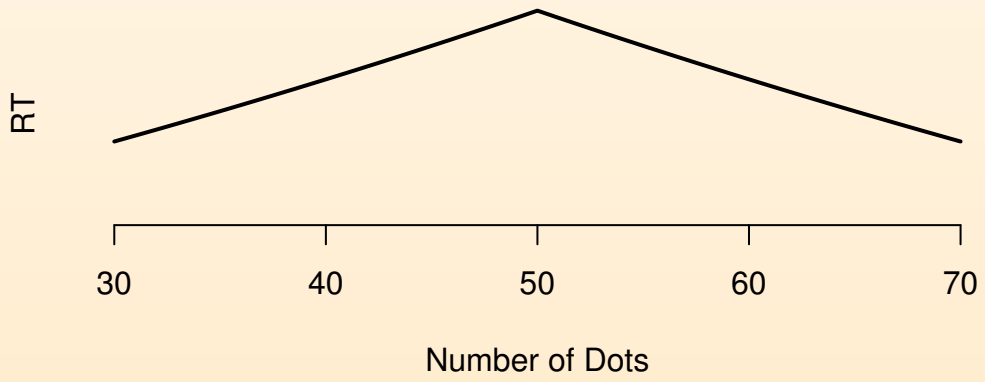
Example 2: A Model for Mean RT

- **The relationship between RT and difficulty.**

- **For the current data, difficulty may be characterized as number of dots on a given trial (close to 50 = difficult; far from 50 = easy).**

Example 2: A Model for Mean RT

- **The relationship between RT and difficulty.**
- **For the current data, difficulty may be characterized as number of dots on a given trial (close to 50 = difficult; far from 50 = easy).**
- **General finding: RT decreases with difficulty.**



Mathematical Functions

- **Two functions describing this relationship:**
  - n **is number of dots**
  - n_0 **is number of dots at which difficulty is highest (50 for the current experiment).**

Mathematical Functions

- **Two functions describing this relationship:**
  - n **is number of dots**
  - n_0 **is number of dots at which difficulty is highest (50 for the current experiment).**
  - **Linear function:**

$$\mathsf{E}(RT) = b + a|n - n_0|$$

Mathematical Functions

- **Two functions describing this relationship:**
  - n **is number of dots**
  - n_0 **is number of dots at which difficulty is highest (50 for the current experiment).**
  - **Linear function:**

$$\mathsf{E}(RT) = b + a|n - n_0|$$

  - **Exponential function:**

$$\mathsf{E}(RT) = b + a\exp(-\lambda|n - n_0|)$$

Mathematical Functions

- **Two functions describing this relationship:**
  - n **is number of dots**
  - n_0 **is number of dots at which difficulty is highest (50 for the current experiment).**
  - **Linear function:**

$$\mathsf{E}(RT) = b + a|n - n_0|$$

  - **Exponential function:**

$$\mathsf{E}(RT) = b + a\exp(-\lambda|n - n_0|)$$

- **Both functions can be modified to incorporate difficulty conditions; participant-level effects.**

Participant & Difficulty Effects

- **Modifying the functions:**
  - **Function for individual i's RT on trial j.**
  - **$I(h_{ij})$ equals 1 when stimulus is hard, 0 when stimulus is easy.**

Participant & Difficulty Effects

- **Modifying the functions:**
  - **Function for individual i's RT on trial j.**
  - **$I(h_{ij})$ equals 1 when stimulus is hard, 0 when stimulus is easy.**
  - **Linear function:**

$$\mathsf{E}(RT_{ij}) = b_i + (a_i + \beta_i I(h_{ij}))|n_{ij} - 50|$$

Participant & Difficulty Effects

- **Modifying the functions:**
  - **Function for individual i's RT on trial j.**
  - **$I(h_{ij})$ equals 1 when stimulus is hard, 0 when stimulus is easy.**
  - **Linear function:**

$$\mathsf{E}(RT_{ij}) = b_i + (a_i + \beta_i I(h_{ij}))|n_{ij} - 50|$$

  - **Exponential function:**

$$\mathsf{E}(RT_{ij}) = b_i + (a_i + \beta_i I(h)) \exp(-(\lambda_i + \gamma_i I(h_{ij}))|n_{ij} - 50|)$$

Participant & Difficulty Effects

- **Modifying the functions:**
  - **Function for individual i's RT on trial j.**
  - **$I(h_{ij})$ equals 1 when stimulus is hard, 0 when stimulus is easy.**
  - **Linear function:**

$$\mathsf{E}(RT_{ij}) = b_i + (a_i + \beta_i I(h_{ij}))|n_{ij} - 50|$$

  - **Exponential function:**

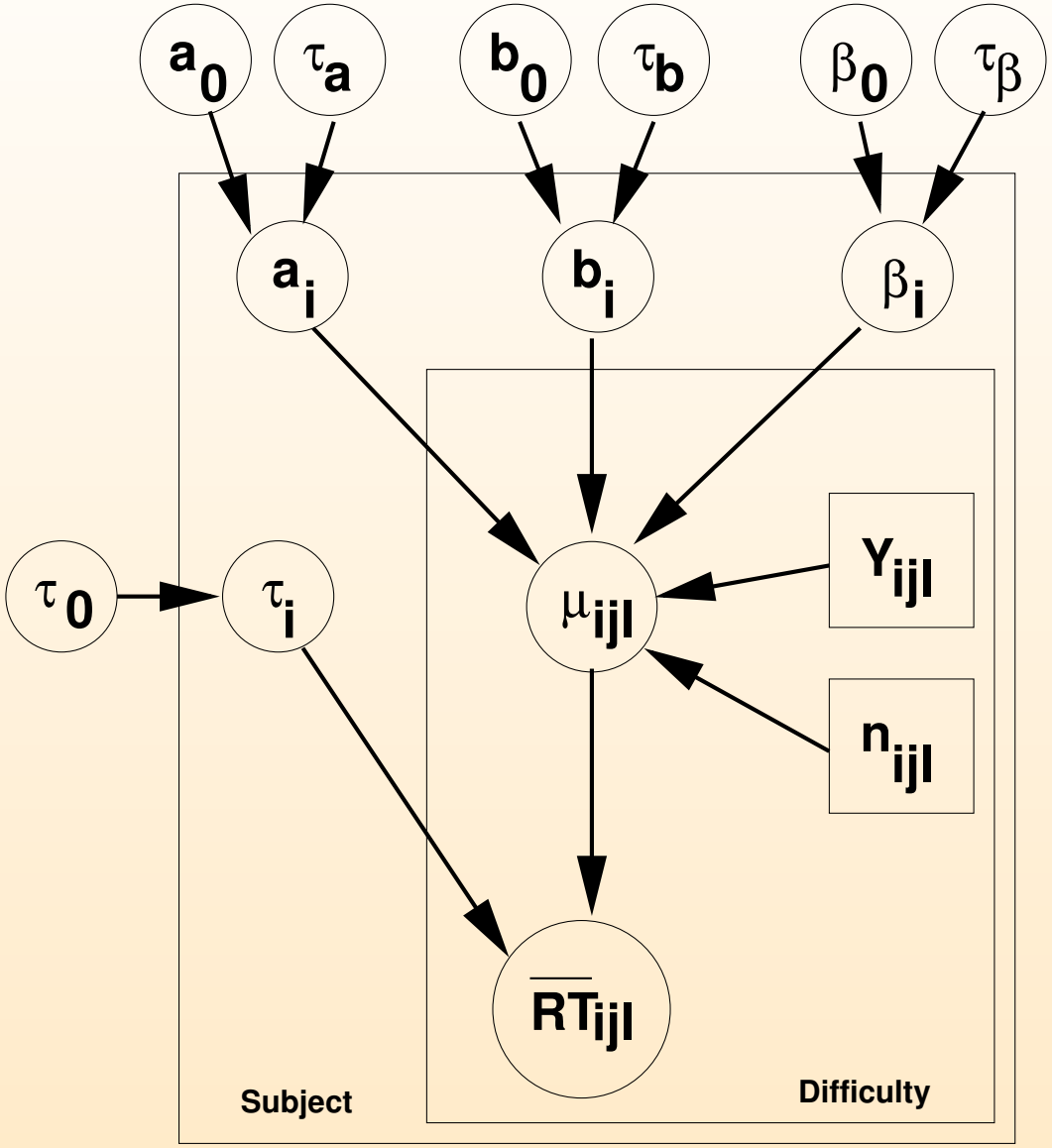
$$\mathsf{E}(RT_{ij}) = b_i + (a_i + \beta_i I(h)) \exp(-(\lambda_i + \gamma_i I(h_{ij}))|n_{ij} - 50|)$$

- **Arrive at a model for RT by taking $RT_{ij} \sim N(\mathsf{E}(RT_{ij}), \tau_i)$.**

Linear Model Specification

```
model
{
for (i in 1:N)  {
  for (j in 1:C) {
  for (l in 1:D) {
    mu[i,j,l] <- b[i] + (a[i] + hard[i,j,l]*beta[i]) *
                 (dots[i,j,l]-50.0)
    RT[i,j,l] ~ dnorm(mu[i,j,l], tau[i])
  }
  }
  b[i] ~ dnorm(b.c,tau.b)
  a[i] ~ dnorm(a.c,tau.a)
  beta[i] ~ dnorm(beta.c,tau.beta)
  tau[i] ~ dgamma(1.0,tau.0)
}
b.c ~ dgamma(1.0,1.0E-3)
a.c ~ dnorm(0.0,1.0E-3)
beta.c ~ dnorm(0.0,1.0E-3)
tau.b ~ dgamma(1.0,1.0E-3)
tau.a ~ dgamma(1.0,1.0E-3)
tau.beta ~ dgamma(1.0,1.0E-3)
tau.0 ~ dgamma(1.0,1.0E-3)
}

list(
b.c=5.0,a.c=1.0,beta.c=-0.5,
tau.b=1.0,tau.a=1.0,tau.beta=1.0,
tau.0=1.0
)
```

Model Doodle



Notes on the Model Specification

- **Number of dots have been divided into D=6 groups of difficulty. For example, group 1 contains RTs for 48-52 dots, group 2 contains RTs for 53-57 dots (and for 43-47 dots), ..., group 6 contains RTs for 73-77 dots (and for 23-27 dots). Examination of the data file should make this clearer.**

Notes on the Model Specification

- **Number of dots have been divided into D=6 groups of difficulty. For example, group 1 contains RTs for 48-52 dots, group 2 contains RTs for 53-57 dots (and for 43-47 dots), ..., group 6 contains RTs for 73-77 dots (and for 23-27 dots). Examination of the data file should make this clearer.**

- **There are C=2 difficulty conditions (hard/easy) that are referred to in the specification.**

Notes on the Model Specification

- Number of dots have been divided into D=6 groups of difficulty. For example, group 1 contains RTs for 48-52 dots, group 2 contains RTs for 53-57 dots (and for 43-47 dots), ..., group 6 contains RTs for 73-77 dots (and for 23-27 dots). Examination of the data file should make this clearer.

- There are C=2 difficulty conditions (hard/easy) that are referred to in the specification.

- This specification references matrices of 3 dimensions. That is, mu[i,j,l] corresponds to the expected RT for subject i in difficulty condition j in dots group l.

Notes on the Model Specification

- Number of dots have been divided into D=6 groups of difficulty. For example, group 1 contains RTs for 48-52 dots, group 2 contains RTs for 53-57 dots (and for 43-47 dots), ..., group 6 contains RTs for 73-77 dots (and for 23-27 dots). Examination of the data file should make this clearer.

- There are C=2 difficulty conditions (hard/easy) that are referred to in the specification.

- This specification references matrices of 3 dimensions. That is, mu[i,j,l] corresponds to the expected RT for subject i in difficulty condition j in dots group l.

- Not all participants have been observed at extreme numbers of dots. Thus, there are some missing values in the response variable RT (denoted by NA). WinBUGS can handle these missing values.

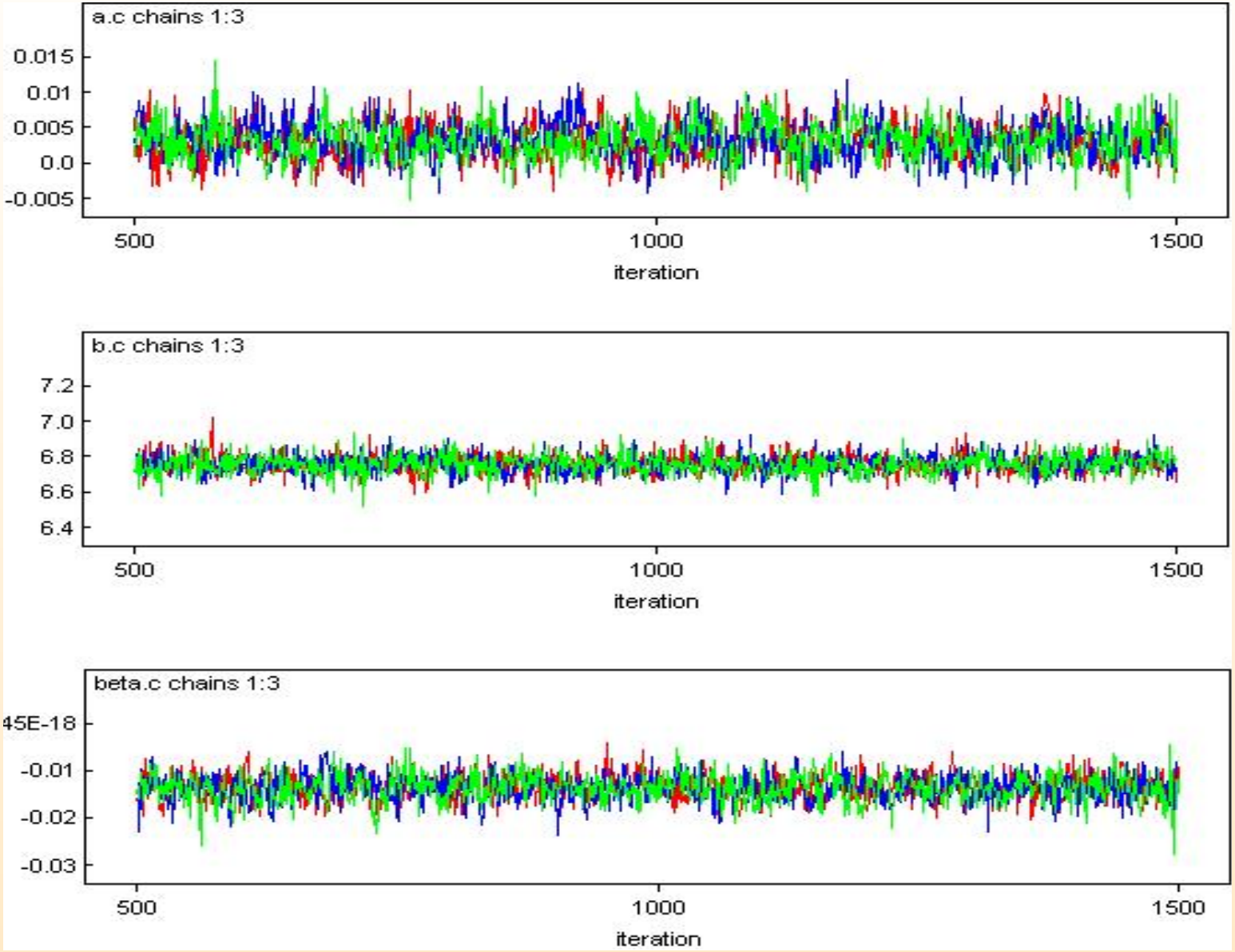
- Sampling from this model is very fast.

Results

- 3 chains sampled for 1500 iterations each; first 500 of each chain discarded.

Results

- **3 chains sampled for 1500 iterations each; first 500 of each chain discarded.**



Results

- **Based on sampled parameters, we obtain 95% posterior intervals:**
  - **b.c: (6.66,6.86)**
  - **a.c: (-.002,.008)**
  - **beta.c: (-.019,-.014)**

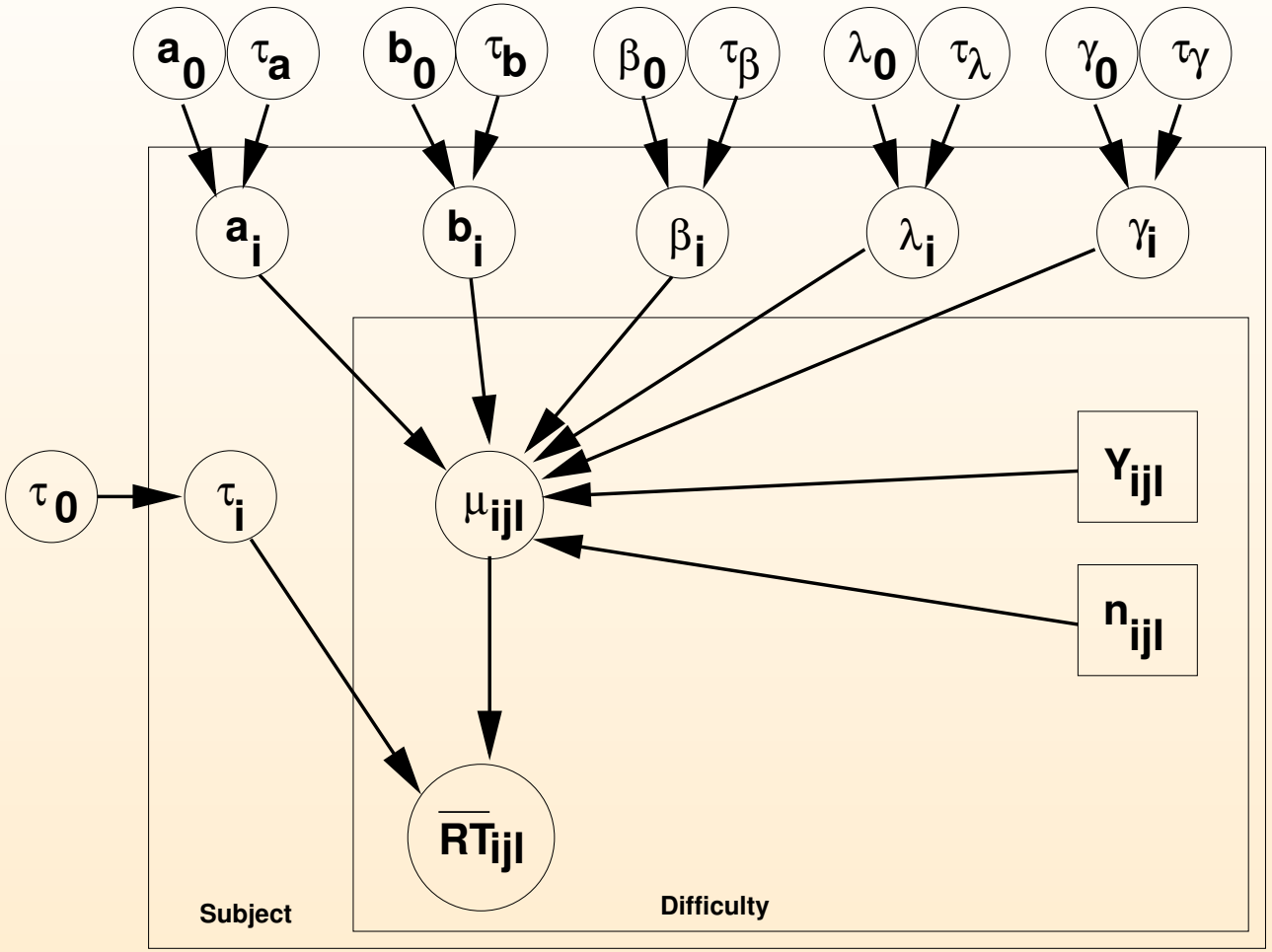
Results

- **Based on sampled parameters, we obtain 95% posterior intervals:**
  - **b.c: (6.66,6.86)**
  - **a.c: (-.002,.008)**
  - **beta.c: (-.019,-.014)**
- **RTs seem to decrease with difficulty in the hard condition but not the easy condition.**
- **Experiment design responsible for this?**

Exponential Model Specification

```
model
{
for (i in 1:N)  {
  for (j in 1:C) {
  for (l in 1:D) {
    mu[i,j,l] <- b[i] + (a[i] + hard[i,j,l]*beta[i]) *
      exp(-(dots[i,j,l]-50.0)*(lambda[i]+hard[i,j,l]*gamma[i]))
    RT[i,j,l] ~ dnorm(mu[i,j,l], tau[i])
  }
  }
  b[i] ~ dnorm(b.c,tau.b)
  a[i] ~ dnorm(a.c,tau.a)
  beta[i] ~ dnorm(beta.c,tau.beta)
  lambda[i] ~ dnorm(lambda.c,tau.lambda)
  gamma[i] ~ dnorm(gamma.c,tau.gamma)
  tau[i] ~ dgamma(1.0,tau.0)
}
b.c ~ dgamma(0.1,0.01)
a.c ~ dnorm(0.0,1.0E-3)
beta.c ~ dnorm(0.0,1.0E-3)
lambda.c ~ dnorm(0.0,1.0E-3)
gamma.c ~ dnorm(0.0,0.01)
tau.b ~ dgamma(0.15,0.01)
tau.a ~ dgamma(1.0,0.01)
tau.beta ~ dgamma(0.3,0.01)
tau.lambda ~ dgamma(4.0,0.01)
tau.gamma ~ dgamma(3.0,0.01)
tau.0 ~ dgamma(1.0,1.0)
}
```

Model Doodle



Starting values

```
list(
b.c=5.0,a.c=1.0,beta.c=-0.5,lambda.c=0.0,gamma.c=0.0,
tau.b=1.0,tau.a=1.0,tau.beta=1.0,tau.lambda=1.0,tau.gamma=1.0,
tau.0=1.0,b = c(5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5), a = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1), beta = c(-0.5, -0.5, -0.5, -0.5, -0.5, -0.5,
-0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5,
-0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5,
-0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5,
-0.5, -0.5, -0.5, -0.5),
lambda = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0), gamma = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0), tau = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1)
)
```

Notes on the Model Specification

- **The exponential specification is similar to the linear specification: RT[i,j,l] refers to the RT of participant i at difficulty level j at dots grouping l.**

Notes on the Model Specification

- **The exponential specification is similar to the linear specification: RT[i,j,l] refers to the RT of participant i at difficulty level j at dots grouping l.**

- **Starting values for the participant-level parameters must be specified; allowing WinBUGS to generate these starting values results in crashes.**

Notes on the Model Specification

- The exponential specification is similar to the linear specification: RT[i,j,l] refers to the RT of participant i at difficulty level j at dots grouping l.

- Starting values for the participant-level parameters must be specified; allowing WinBUGS to generate these starting values results in crashes.

- Some of the simulated parameters have fairly high autocorrelations from iteration to iteration. Possible remedies include running the model for more iterations, giving each chain different starting values, or checking the "over relax" button on the Update menu.

Notes on the Model Specification

- The exponential specification is similar to the linear specification: RT[i,j,l] refers to the RT of participant i at difficulty level j at dots grouping l.

- Starting values for the participant-level parameters must be specified; allowing WinBUGS to generate these starting values results in crashes.

- Some of the simulated parameters have fairly high autocorrelations from iteration to iteration. Possible remedies include running the model for more iterations, giving each chain different starting values, or checking the "over relax" button on the Update menu.

- We have attempted to make the prior distributions for model parameters noninformative, while still keeping the density in the ballpark of the parameter estimates. We obtained similar parameter estimates for a number of noninformative prior distributions.

Notes on the Model Specification

- The exponential specification is similar to the linear specification: RT[i,j,l] refers to the RT of participant i at difficulty level j at dots grouping l.
- Starting values for the participant-level parameters must be specified; allowing WinBUGS to generate these starting values results in crashes.
- Some of the simulated parameters have fairly high autocorrelations from iteration to iteration. Possible remedies include running the model for more iterations, giving each chain different starting values, or checking the "over relax" button on the Update menu.
- We have attempted to make the prior distributions for model parameters noninformative, while still keeping the density in the ballpark of the parameter estimates. We obtained similar parameter estimates for a number of noninformative prior distributions.
- The default sampling method for this model requires 4000 iterations before parameter summaries are allowed. After the first 4000 iterations, all WinBUGS tools are available to you.

Results

- **3 chains of parameters sampled for 7000 iterations each; first 6000 of each chain were discarded.**

Results

- **3 chains of parameters sampled for 7000 iterations each; first 6000 of each chain were discarded.**

- **Autocorrelations are relatively high, but other plots/statistics give evidence of convergence.**

Results

- 3 chains of parameters sampled for 7000 iterations each; first 6000 of each chain were discarded.

- Autocorrelations are relatively high, but other plots/statistics give evidence of convergence.

- 95% posterior intervals:
  - b.c: (6.43,6.70)
  - a.c: (0.21,0.43)
  - beta.c: (-0.34,-0.19)
  - lambda.c: (0.01,0.06)
  - gamma.c: (-0.03,0.06)

Results

- 3 chains of parameters sampled for 7000 iterations each; first 6000 of each chain were discarded.

- Autocorrelations are relatively high, but other plots/statistics give evidence of convergence.

- 95% posterior intervals:
  - b.c: (6.43,6.70)
  - a.c: (0.21,0.43)
  - beta.c: (-0.34,-0.19)
  - lambda.c: (0.01,0.06)
  - gamma.c: (-0.03,0.06)

- Intervals lead to similar conclusions as for the linear model.

Comparing the Models

- **Which model is better: linear or exponential?**

Comparing the Models

- **Which model is better: linear or exponential?**
- **WinBUGS can calculate the DIC statistic, which is a measure of model complexity and can be used for model comparison.**
- **Model with smaller DIC is taken to be better.**

Comparing the Models

- **Which model is better: linear or exponential?**
- **WinBUGS can calculate the DIC statistic, which is a measure of model complexity and can be used for model comparison.**
- **Model with smaller DIC is taken to be better.**
- **The DIC Tool yields -155.25 and -375.02 for linear and exponential models, respectively.**

Comparing the Models

- **Which model is better: linear or exponential?**
- **WinBUGS can calculate the DIC statistic, which is a measure of model complexity and can be used for model comparison.**
- **Model with smaller DIC is taken to be better.**
- **The DIC Tool yields -155.25 and -375.02 for linear and exponential models, respectively.**
- **Based on this statistic, we conclude that the exponential model is better.**

Example 3: Estimating RT Distributions

- **Aside from modeling mean RT (as in Example 2), it is also of interest to model RT distributions.**

Example 3: Estimating RT Distributions

- **Aside from modeling mean RT (as in Example 2), it is also of interest to model RT distributions.**

- **Rouder et al. (2003) use the Weibull to model RT distributions.**

- **If y_{ij} is participant i's RT on trial j, we can write the density as:**

$$f(y_{ij} \mid \psi_i, \theta_i, \beta_i) = \frac{\beta_i (y_{ij} - \psi_i)^{\beta_i - 1}}{\theta_i^{\beta_i}} \exp\left[-\frac{(y_{ij} - \psi_i)^{\beta_i}}{\theta_i^{\beta_i}} \right], \tag{1}$$

for $y_{ij} > \psi_i$.

Example 3: Estimating RT Distributions

- **Aside from modeling mean RT (as in Example 2), it is also of interest to model RT distributions.**

- **Rouder et al. (2003) use the Weibull to model RT distributions.**

- **If y_{ij} is participant i's RT on trial j, we can write the density as:**

$$f(y_{ij} \mid \psi_i, \theta_i, \beta_i) = \frac{\beta_i (y_{ij} - \psi_i)^{\beta_i - 1}}{\theta_i^{\beta_i}} \exp\left[-\frac{(y_{ij} - \psi_i)^{\beta_i}}{\theta_i^{\beta_i}} \right], \tag{1}$$

  for $y_{ij} > \psi_i$.

- **ψ, θ, and β parameters correspond to the distribution's shift, scale, and shape, respectively.**

Hierarchical Structure

- **Model is hierarchical:**
  - **Each β_i assumed to arise from a common Gamma(η_1, η_2) distribution.**

Hierarchical Structure

- **Model is hierarchical:**
  - **Each β_i assumed to arise from a common Gamma(η_1, η_2) distribution.**
  - **Given β_i, each $\theta_i^{\beta_i}$ assumed to arise from a common Inverse Gamma(ξ_1, ξ_2) distribution.**

Hierarchical Structure

- **Model is hierarchical:**
  - **Each β_i assumed to arise from a common Gamma(η_1, η_2) distribution.**
  - **Given β_i, each $\theta_i^{\beta_i}$ assumed to arise from a common Inverse Gamma(ξ_1, ξ_2) distribution.**
  - **Each ψ_i assumed to arise from a uniform distribution.**

Hierarchical Structure

- **Model is hierarchical:**
  - **Each β_i assumed to arise from a common Gamma(η_1, η_2) distribution.**
  - **Given β_i, each $\theta_i^{\beta_i}$ assumed to arise from a common Inverse Gamma(ξ_1, ξ_2) distribution.**
  - **Each ψ_i assumed to arise from a uniform distribution.**

- **All hyperparameters ($\eta_1, \eta_2, \xi_1, \xi_2$) are given Gamma prior distributions.**

Hierarchical Structure

- **Model is hierarchical:**
  - **Each β_i assumed to arise from a common Gamma(η_1, η_2) distribution.**
  - **Given β_i, each $\theta_i^{\beta_i}$ assumed to arise from a common Inverse Gamma(ξ_1, ξ_2) distribution.**
  - **Each ψ_i assumed to arise from a uniform distribution.**

- **All hyperparameters ($\eta_1, \eta_2, \xi_1, \xi_2$) are given Gamma prior distributions.**

- **The resulting model estimates RT distributions at participant level, and hyperparameters can tell us how parameters vary across participants.**

The Zeros Trick

- **Model implementation is difficult because not all distributions are predefined in WinBUGS.**

The Zeros Trick

- **Model implementation is difficult because not all distributions are predefined in WinBUGS.**

- **Must rely on the "zeros trick" for specifying new distributions.**
  - **If x has a Poisson distribution and $x = 0$, the Poisson density reduces to $\exp^{-\lambda}$.**

The Zeros Trick

- **Model implementation is difficult because not all distributions are predefined in WinBUGS.**

- **Must rely on the "zeros trick" for specifying new distributions.**
  - **If x has a Poisson distribution and $x = 0$, the Poisson density reduces to $\exp^{-\lambda}$.**
  - **We can get a new distribution into WinBUGS by setting up dummy data of 0's and setting λ to be the negative log-likelihood of the new distribution.**

The Zeros Trick

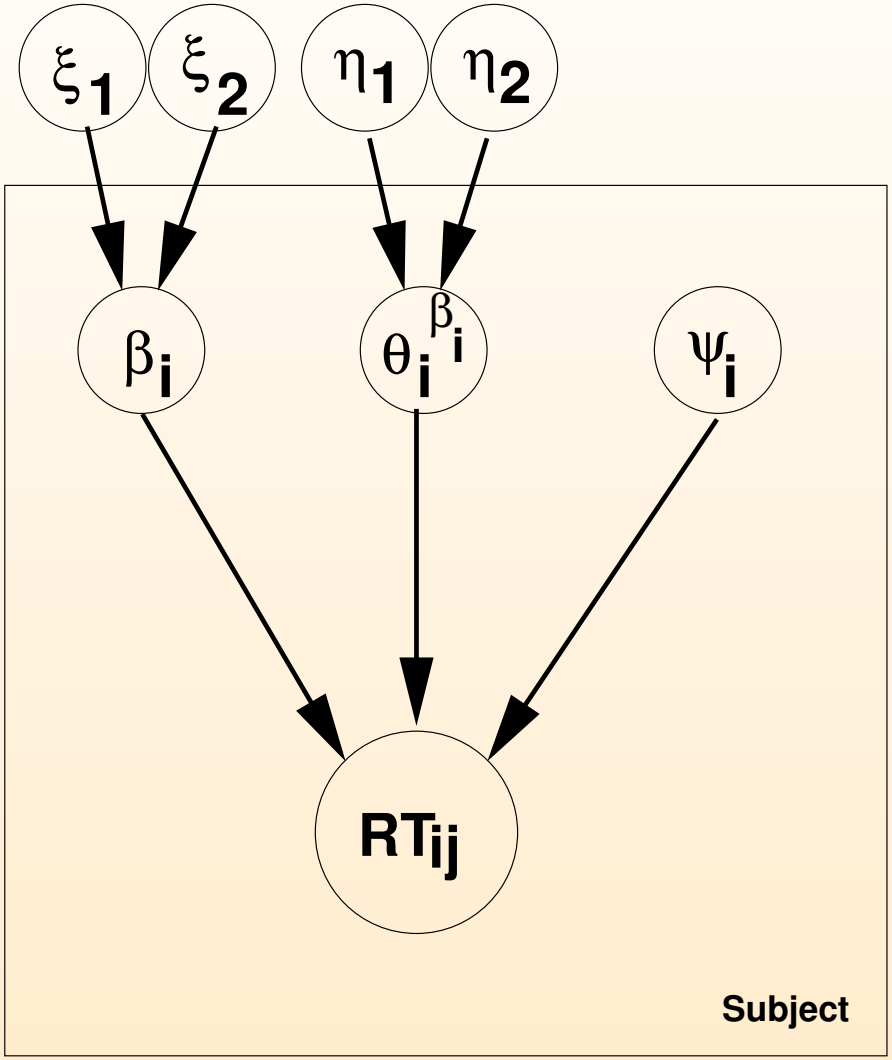
- **Model implementation is difficult because not all distributions are predefined in WinBUGS.**

- **Must rely on the "zeros trick" for specifying new distributions.**
  - **If x has a Poisson distribution and $x = 0$, the Poisson density reduces to $\exp^{-\lambda}$.**
  - **We can get a new distribution into WinBUGS by setting up dummy data of 0's and setting λ to be the negative log-likelihood of the new distribution.**
  - **A constant is added to λ to ensure $\lambda > 0$.**

Model Specification

```
model
{
# Works if you change "real nonlinear" to "UpdaterSlice" in methods.odc
# y[i,j] is RT of participant i on trial j
# Only RTs from easy condition are used
# minrt is minimum rt for each participant
C <- 100000000
for (i in 1:N){
  for (j in 1:trials){
    zeros[i,j] <- 0
    term1[i,j] <- beta[i]*log(theta[i]) + pow(y[i,j] - psi[i],
                  beta[i])/pow(theta[i],beta[i])
    term2[i,j] <- log(beta[i]) + (beta[i]-1)*log(y[i,j] -
                  psi[i])
    phi[i,j] <- term1[i,j] - term2[i,j] + C
    zeros[i,j] ~ dpois(phi[i,j])
  }
beta[i] ~ dgamma(eta1,eta2)I(0.01,)
zero[i] <- 0
theta[i] ~ dflat()
phip[i] <- (xi1 + 1)*log(pow(theta[i],beta[i])) +
            xi2/pow(theta[i],beta[i]) + loggam(xi1) - xi1*log(xi2)
zero[i] ~ dpois(phip[i])
psi[i] ~ dunif(0,minrt[i])
}
# Priors as recommended by Rouder et al.
eta1 ~ dgamma(2,0.02)
eta2 ~ dgamma(2,0.04)
xi1 ~ dgamma(2,0.1)
xi2 ~ dgamma(2,2.85)
}
```

Model Doodle



Starting values

```
list(beta = c(1.15,1.3,1.27,1.04,1.16,1.18,1.34,
1.47,1.16,1.22,1.4,1.18,1.08,1.14,1.13,1.13,1.58,1.55,
1.5,1.55,1.3,1.34,1.28,1.22,1.37,0.9,1.26,1.47,1.64,
1.16,1.15,1.39,1.09,1.35,1.69,1.45,1.23), eta1 = 71.5,
eta2 = 56.12, psi = c(542.44,397.88,631.67,397.8,468.62,
654.39,528.16,435.96,562.58,699.9,614.47,400.18,291.57,
651.68,564.97,364.07,416.67,199.92,232.89,525.56,336.01,
421.95,562.76,414.89,397.26,205.09,378.95,309,241.95,
309.44,638.07,234.44,586.83,337.13,346.51,548.37,442.76
), theta = c(1263.77,589.18,763.83,705.66,1655.92,925.8,
1624.29,452.35,650.88,874.17,1164.92,468.03,1221.61,
938.44,740.69,1023,354.17,756.69,565.12,1481.56,1160.22,
917.63,1032.16,1634.01,1118.65,185.36,694.14,381.59,
498.82,755.39,1326.63,1421.59,1096.91,581,490.94,1134.19,
1285.15), xi1 = 0.12, xi2 = 3.92)
```

Notes on the Model Specification

- **The Weibull model (with shift parameter) is implemented in WinBUGS via the zeros trick.**

Notes on the Model Specification

- **The Weibull model (with shift parameter) is implemented in WinBUGS via the zeros trick.**

- **We cannot specify a prior distribution for $\theta_i^{\beta_i}$, because WinBUGS will not accept prior distributions for functions of two parameters. Instead, we specify the Inverse Gamma distribution as a function of only θ_i by again using the zeros trick.**

Notes on the Model Specification

- **The Weibull model (with shift parameter) is implemented in WinBUGS via the zeros trick.**

- **We cannot specify a prior distribution for $\theta_i^{\beta_i}$, because WinBUGS will not accept prior distributions for functions of two parameters. Instead, we specify the Inverse Gamma distribution as a function of only θ_i by again using the zeros trick.**

- **The zeros trick requires negative log-likelihoods of our distributions. That is why the forms of our new distributions look so strange in the implementation.**

Notes on the Model Specification

- **The Weibull model (with shift parameter) is implemented in WinBUGS via the zeros trick.**

- **We cannot specify a prior distribution for $\theta_i^{\beta_i}$, because WinBUGS will not accept prior distributions for functions of two parameters. Instead, we specify the Inverse Gamma distribution as a function of only θ_i by again using the zeros trick.**

- **The zeros trick requires negative log-likelihoods of our distributions. That is why the forms of our new distributions look so strange in the implementation.**

- **Running this model using WinBUGS defaults always results in error messages around iteration 1100. We could not figure out why this happens, but the WinBUGS authors suggest that this problem can sometimes be avoided by changing sampling methods. These sampling methods are found in the file Updater/Rsrc/Methods.odc; by changing the method listed in the code, the model usually runs without errors.**

- **The first few (say, 50) iterations take a long time (at least a few minutes) to run, longer if you have multiple chains. The subsequent iterations are faster, taking 3-5 minutes per 1000 iterations.**

Results

- **3 chains of parameters sampled for 4000 iterations each; first 1000 iterations were discarded.**

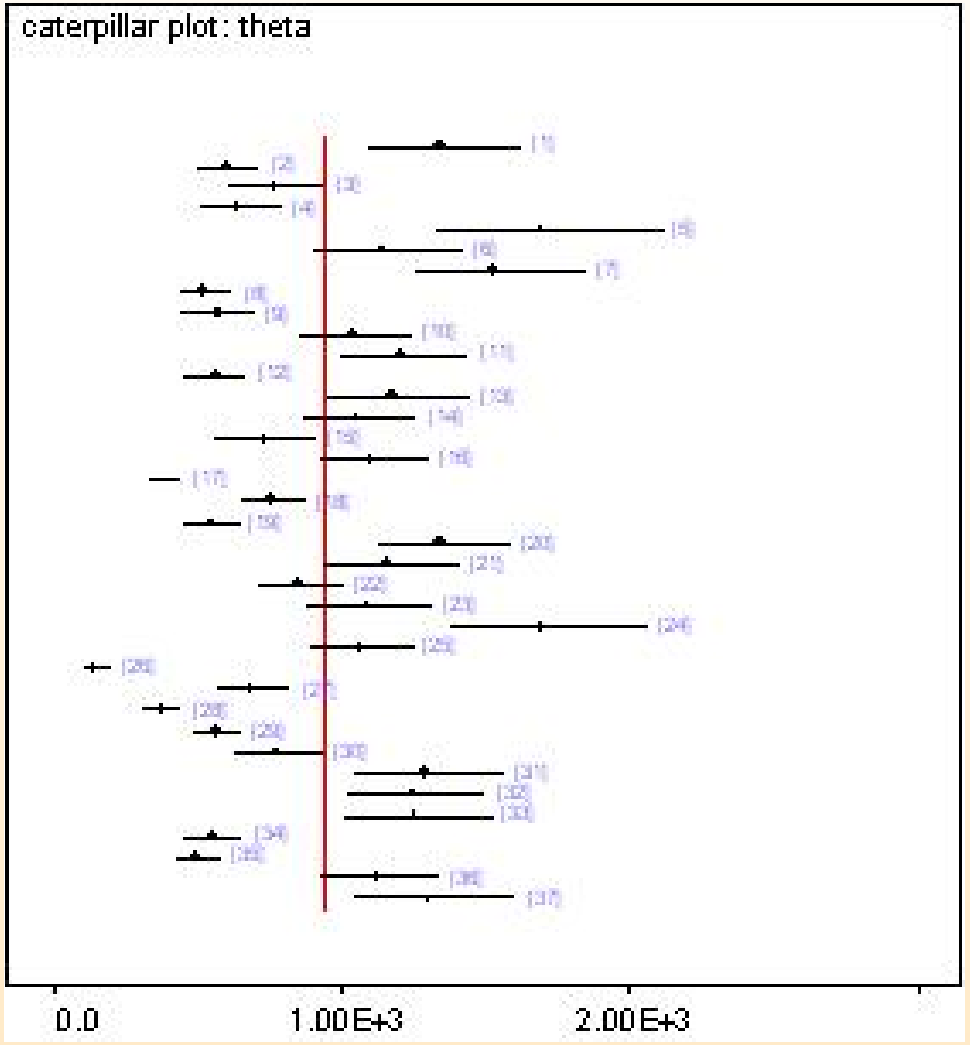
Results

- **3 chains of parameters sampled for 4000 iterations each; first 1000 iterations were discarded.**

- **Plots/statistics in WinBUGS signify convergence.**

Caterpillar Plots

Participant-level parameters can be visualized using caterpillar plots in the Comparison Tool:



Miscellaneous Notes

- **In the provided set of probability distributions, WinBUGS measures scale parameters by their precision (which equals 1/variance). For example, to assign a normal distribution with mean 1 and variance 2, you would type "dnorm(1,0.5)". It is a good idea to check the parameterization of WinBUGS distributions, which can be found in the help files.**

Miscellaneous Notes

- **In the provided set of probability distributions, WinBUGS measures scale parameters by their precision (which equals 1/variance). For example, to assign a normal distribution with mean 1 and variance 2, you would type "dnorm(1,0.5)". It is a good idea to check the parameterization of WinBUGS distributions, which can be found in the help files.**

- **As parameters are simulated for more iterations, parameter estimates become more accurate. It is often the case, however, that people report parameter estimates to more decimal places than are justified. The "MC error" values give you an idea of how many decimal places you can accurately report.**