

# Brief WinBUGS tutorial

By

Hedibert Freitas Lopes  
Graduate School of Business  
University of Chicago

# Nonlinear growth curve

Carlin and Gelfand (1991) present a nonconjugate Bayesian analysis of the following data set from Ratkowsky (1983):

Dugong (sea cows)	1	2	3	...	26	27
Age (X)	1.00	1.50	1.50	...	29.0	31.50
Length (Y)	1.80	1.85	1.87	...	2.27	2.57

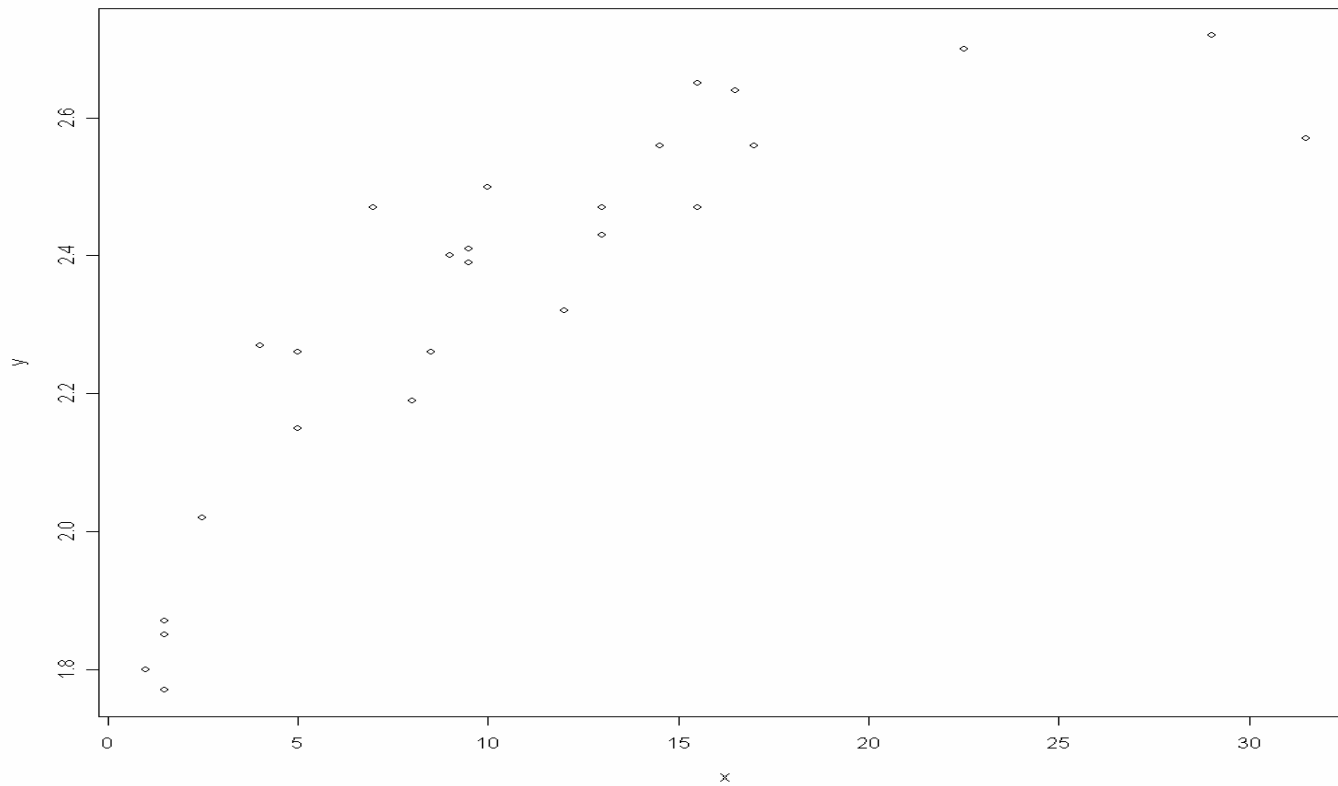
Carlin and Gelfand (1991) model this data using a nonlinear growth curve with no inflection point and an asymptote as  $x_i$  tends to infinity:

$$y_i \sim N(\mu_i, \tau^{-1})$$
$$\mu_i = \alpha - \beta\gamma^{x_i}$$

for  $i = 1, \dots, 27$ ,  $\alpha, \beta > 1$  and  $0 < \gamma < 1$ .

Standard noninformative priors are adopted for  $\alpha, \beta$  and  $\tau$ , and a uniform prior on  $(0,1)$  is assumed for  $\gamma$ .

	y	x		y	x		y	x		y	x
1	1.80	1.0	8	2.26	5.0	15	2.50	10.0	22	2.47	15.5
2	1.85	1.5	9	2.47	7.0	16	2.32	12.0	23	2.64	16.5
3	1.87	1.5	10	2.19	8.0	17	2.32	12.0	24	2.56	17.0
4	1.77	1.5	11	2.26	8.5	18	2.43	13.0	25	2.70	22.5
5	2.02	2.5	12	2.40	9.0	19	2.47	13.0	26	2.72	29.0
6	2.27	4.0	13	2.39	9.5	20	2.56	14.5	27	2.57	31.5
7	2.15	5.0	14	2.41	9.5	21	2.65	15.5			



# WinBUGS code

```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha - beta * pow(gamma,x[i])
  }
  alpha ~ dnorm(0.0, 1.0E-6)
  beta ~ dnorm(0.0, 1.0E-6)
  gamma ~ dunif(0.0, 1.0)
  tau ~ dgamma(0.01, 0.01)
}
```

# Data and initial values

## DATA

```
list(x=c(1.0,1.5,1.5,1.5,2.5,4.0,5.0,5.0,7.0,8.0,  
8.5,9.0,9.5,9.5,10.0,12.0,12.0,13.0,13.0,14.5,  
15.5,15.5,16.5,17.0,22.5,29.0,31.5),y=c(1.80,  
1.85,1.87,1.77,2.02,2.27,2.15,2.26,2.47,2.19,  
2.26,2.40,2.39,2.41,2.50,2.32,2.32,2.43,2.47,  
2.56,2.65,2.47,2.64,2.56,2.70,2.72,2.57),N=27)
```

## FIRST INITIAL VALUES

```
list(alpha=1,beta=1,tau=1,gamma=0.9)
```

SECOND INITIAL VALUES ARE RANDOMLY DRAWN FROM  
THE PRIOR DISTRIBUTIONS.

# Model>Specification

The screenshot displays the WinBUGS 14 software interface. The main window is titled "nonlinearregression" and contains the following code:

```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha-beta*x[i]
  }
  alpha ~ dnorm(0.0, 1.0E-6)
  beta ~ dnorm(0.0, 1.0E-6)
  gamma ~ dunif(0.0, 1.0)
  tau ~ dgamma(0.01, 0.01)
}
```

DATA

```
list(x=c(1.0,1.5,1.5,1.5,2.5,4.0,5.0,5.0,7.0,8.0,
8.5,9.0,9.5,9.5,10.0,12.0,12.0,13.0,13.0,14.5,
15.5,15.5,16.5,17.0,22.5,29.0,31.5),y=c(1.80,
1.85,1.87,1.77,2.02,2.27,2.15,2.26,2.47,2.19,
2.26,2.40,2.39,2.41,2.50,2.32,2.32,2.43,2.47,
2.56,2.65,2.47,2.64,2.56,2.70,2.72,2.57),N=27)
```

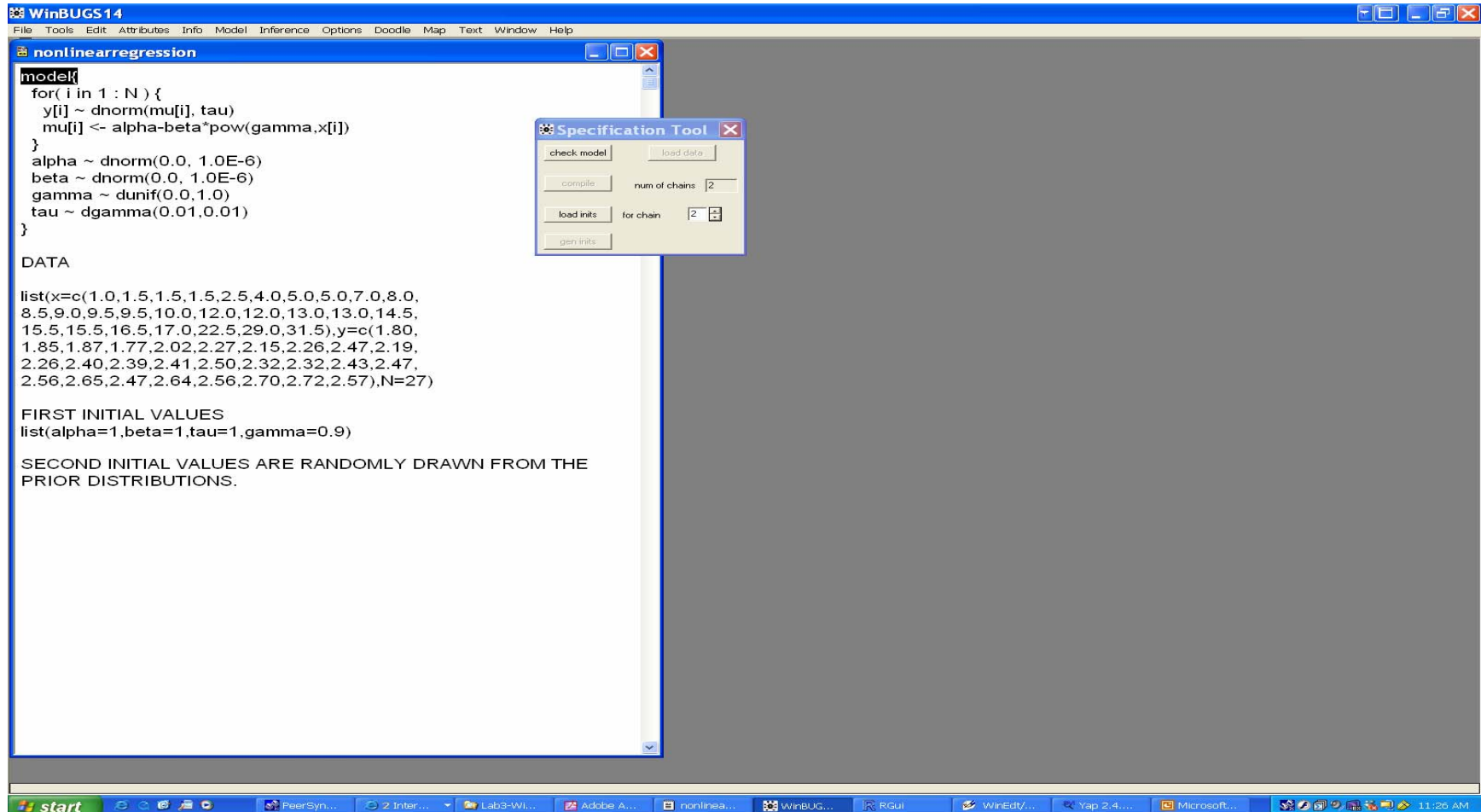
FIRST INITIAL VALUES

```
list(alpha=1,beta=1,tau=1,gamma=0.9)
```

SECOND INITIAL VALUES ARE RANDOMLY DRAWN FROM THE PRIOR DISTRIBUTIONS.

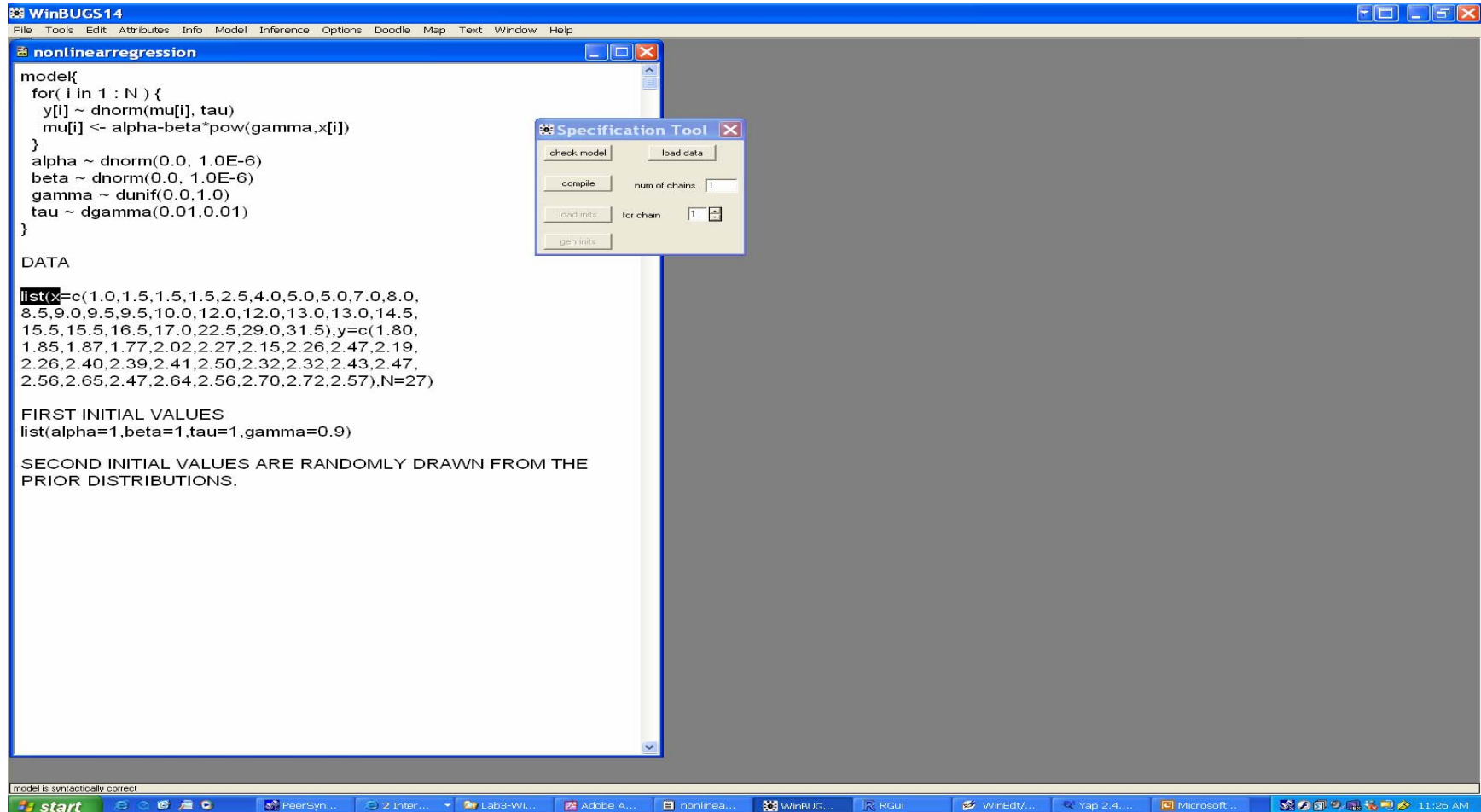
The interface includes a menu bar with options: File, Tools, Edit, Attributes, Info, Model, Inference, Options, Doodle, Map, Text, Window, Help. A context menu is open over the code editor, showing options: Specification..., Update..., Monitor Met, Save State, Seed..., and Script. The Windows taskbar at the bottom shows the Start button and several open applications, including PeerSync P..., 2 Intern..., Lab3-Win..., Adobe Acr..., nonlinearre..., WinBUGS.14, RGui, WinEdt/Mi..., and Yap 2.4.1B..., with the system clock showing 11:25 AM.

# Select model, then “check model”



Bottom message: “model is syntactically correct”

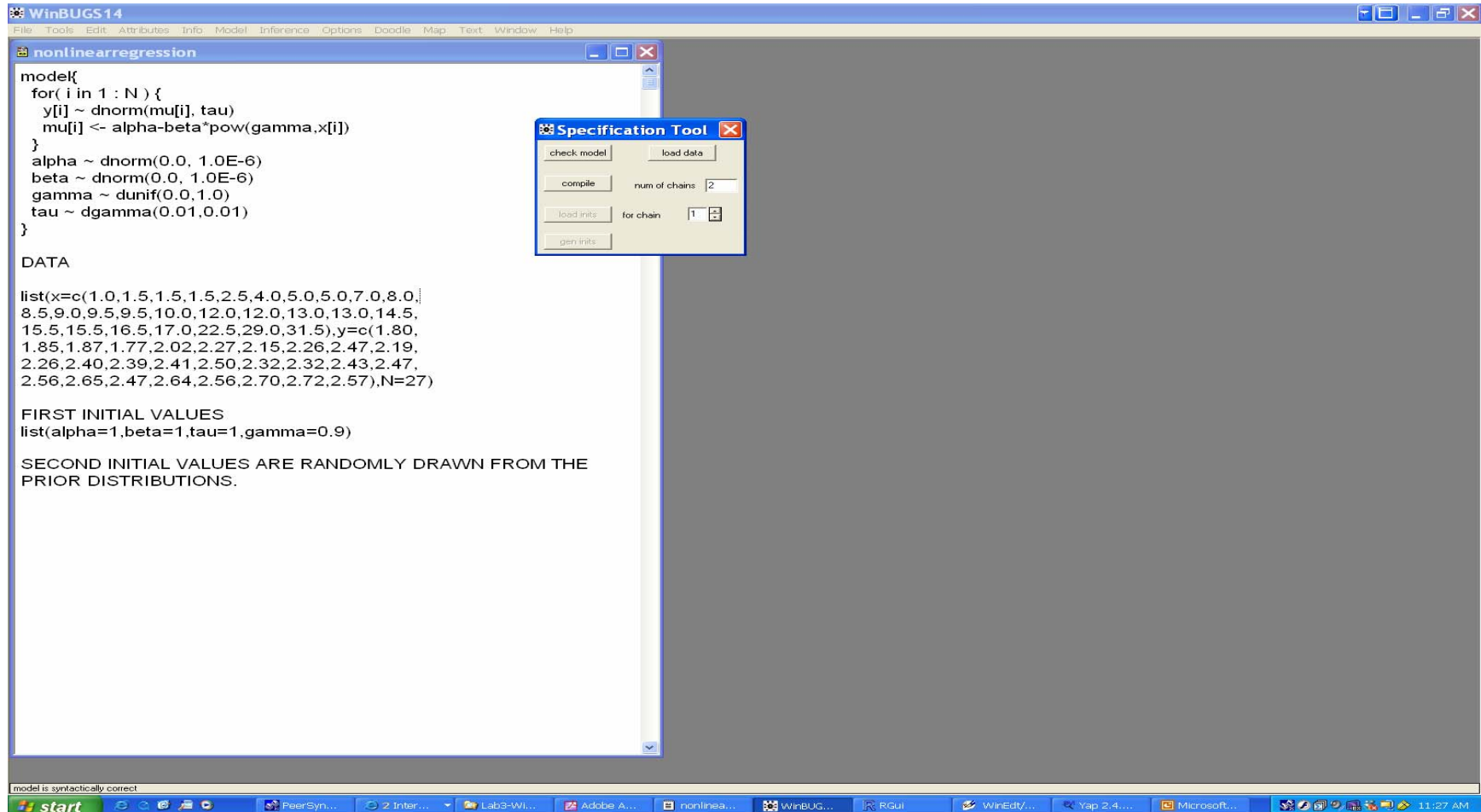
# Select data, then “load data”



Bottom message: “data loaded”



# Set “run of chains” to 2, then “compile” your model.



Bottom message: “model compiled”.

# “load inits” for the 1<sup>st</sup> chain.

The screenshot shows the WinBUGS14 interface. The main window displays the following code and data:

```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha-beta*pow(gamma,x[i])
  }
  alpha ~ dnorm(0.0, 1.0E-6)
  beta ~ dnorm(0.0, 1.0E-6)
  gamma ~ dunif(0.0,1.0)
  tau ~ dgamma(0.01,0.01)
}
```

DATA

```
list(x=c(1.0,1.5,1.5,1.5,2.5,4.0,5.0,5.0,7.0,8.0,
8.5,9.0,9.5,9.5,10.0,12.0,12.0,13.0,13.0,14.5,
15.5,15.5,16.5,17.0,22.5,29.0,31.5),y=c(1.80,
1.85,1.87,1.77,2.02,2.27,2.15,2.26,2.47,2.19,
2.26,2.40,2.39,2.41,2.50,2.32,2.32,2.43,2.47,
2.56,2.65,2.47,2.64,2.56,2.70,2.72,2.57),N=27)
```

FIRST INITIAL VALUES  
`list(alpha=1,beta=1,tau=1,gamma=0.9)`

SECOND INITIAL VALUES ARE RANDOMLY DRAWN FROM THE PRIOR DISTRIBUTIONS.

The Specification Tool dialog box is open, showing the following options:

- check model
- load data
- compile
- num of chains: 2
- load inits for chain: 1
- gen inits

The status bar at the bottom of the WinBUGS window displays the message: "model compiled".

Bottom message: “chain initialized but other chain(s) contain uninitialized variables”.

# “gen inits” for the 2<sup>nd</sup> chain.

The screenshot shows the WinBUGS14 interface. The main window displays the following code and data:

```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha-beta*pow(gamma,x[i])
  }
  alpha ~ dnorm(0.0, 1.0E-6)
  beta ~ dnorm(0.0, 1.0E-6)
  gamma ~ dunif(0.0,1.0)
  tau ~ dgamma(0.01,0.01)
}
```

DATA

```
list(x=c(1.0,1.5,1.5,1.5,2.5,4.0,5.0,5.0,7.0,8.0,
8.5,9.0,9.5,9.5,10.0,12.0,12.0,13.0,13.0,14.5,
15.5,15.5,16.5,17.0,22.5,29.0,31.5),y=c(1.80,
1.85,1.87,1.77,2.02,2.27,2.15,2.26,2.47,2.19,
2.26,2.40,2.39,2.41,2.50,2.32,2.32,2.43,2.47,
2.56,2.65,2.47,2.64,2.56,2.70,2.72,2.57),N=27)
```

FIRST INITIAL VALUES  
`list(alpha=1,beta=1,tau=1,gamma=0.9)`

SECOND INITIAL VALUES ARE RANDOMLY DRAWN FROM THE PRIOR DISTRIBUTIONS.

The Specification Tool dialog box is open, showing the following options:

- check model
- load data
- compile
- num of chains: 2
- load inits
- for chain: 1
- gen inits

The status bar at the bottom of the WinBUGS window displays the message: "initial values generated, model initialized".

Bottom message: “initial values generated, model initialized”.

# Model > Update

The screenshot displays the WinBUGS 14 software interface. The main window shows a model specification for a nonlinear regression model. A context menu is open over the model code, with the 'Update...' option selected. A 'Specification Tool' dialog box is also visible, containing various controls for model execution.

```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha-beta*x[i]
  }
  alpha ~ dnorm(0.0, 1.0E-6)
  beta ~ dnorm(0.0, 1.0E-6)
  gamma ~ dunif(0.0,1.0)
  tau ~ dgamma(0.01,0.01)
}
```

DATA

```
list(x=c(1.0,1.5,1.5,1.5,2.5,4.0,5.0,5.0,7.0,8.0,
8.5,9.0,9.5,9.5,10.0,12.0,12.0,13.0,13.0,14.5,
15.5,15.5,16.5,17.0,22.5,29.0,31.5),y=c(1.80,
1.85,1.87,1.77,2.02,2.27,2.15,2.26,2.47,2.19,
2.26,2.40,2.39,2.41,2.50,2.32,2.32,2.43,2.47,
2.56,2.65,2.47,2.64,2.56,2.70,2.72,2.57),N=27)
```

FIRST INITIAL VALUES  
`list(alpha=1,beta=1,tau=1,gamma=0.9)`

SECOND INITIAL VALUES ARE RANDOMLY DRAWN FROM THE PRIOR DISTRIBUTIONS.

**Specification Tool**

check model    load data

compile    num of chains 2

load inits    for chain 2

gen inits

initial values generated, model initialized

# Let us run the MCMC for 10000 iterations

The screenshot displays the WinBUGS14 software interface. The main window, titled "nonlinearregression", contains the following model code:

```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha-beta*pow(gamma,x[i])
  }
  alpha ~ dnorm(0.0, 1.0E-6)
  beta ~ dnorm(0.0, 1.0E-6)
  gamma ~ dunif(0.0,1.0)
  tau ~ dgamma(0.01,0.01)
}
```

DATA

list(x=c(1.0,1.5,1.5,1.5,2.5,4.0,5.0,5.0,7.0,8.0,8.5,9.0,9.5,9.5,10.0,12.0,12.0,13.0,13.0,14.5,15.5,15.5,16.5,17.0,22.5,29.0,31.5),y=c(1.80,1.85,1.87,1.77,2.02,2.27,2.15,2.26,2.47,2.19,2.26,2.40,2.39,2.41,2.50,2.32,2.32,2.43,2.47,2.56,2.65,2.47,2.64,2.56,2.70,2.72,2.57),N=27)

FIRST INITIAL VALUES  
list(alpha=1,beta=1,tau=1,gamma=0.9)

SECOND INITIAL VALUES ARE RANDOMLY DRAWN FROM THE PRIOR DISTRIBUTIONS.

Two tool windows are visible:

- Specification Tool:** Includes buttons for "check model", "load data", "compile", "load inits", and "gen inits". It also features a "num of chains" field set to 2 and a "for chain" field set to 2.
- Update Tool:** Includes a "updates" field set to 10000, a "refresh" field set to 100, and an "iteration" field set to 0. It also has checkboxes for "over relax" (unchecked) and "adapting" (checked).

The Windows taskbar at the bottom shows the Start button and several open applications, including PeerSync, Internet Explorer, Lab3-Win..., Adobe Acrobat, WinBUGS14, RGui, WinEdt/Mi..., Yap 2.4.18..., and Microsoft P... The system clock indicates 11:35 AM.

# Inference>sample

The screenshot displays the WinBUGS 14 interface. The main window, titled "nonlinearregression", contains the following model code:

```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha-beta*pow(t, gamma)
  }
  alpha ~ dnorm(0.0, 1.0E-6)
  beta ~ dnorm(0.0, 1.0E-6)
  gamma ~ dunif(0.0,1.0)
  tau ~ dgamma(0.01,0.01)
}
```

DATA

```
list(x=c(1.0,1.5,1.5,1.5,2.5,4.0,5.0,5.0,7.0,8.0,
8.5,9.0,9.5,9.5,10.0,12.0,12.0,13.0,13.0,14.5,
15.5,15.5,16.5,17.0,22.5,29.0,31.5),y=c(1.80,
1.85,1.87,1.77,2.02,2.27,2.15,2.26,2.47,2.19,
2.26,2.40,2.39,2.41,2.50,2.32,2.32,2.43,2.47,
2.56,2.65,2.47,2.64,2.56,2.70,2.72,2.57),N=27)
```

FIRST INITIAL VALUES  
`list(alpha=1,beta=1,tau=1,gamma=0.9)`

SECOND INITIAL VALUES ARE RANDOMLY DRAWN FROM THE PRIOR DISTRIBUTIONS.

Two tool windows are open:

- Specification Tool:** Includes buttons for "check model", "load data", "compile", "load inits", and "gen inits". It also has input fields for "num of chains" (set to 2) and "for chain" (set to 2).
- Update Tool:** Includes input fields for "updates" (10000), "refresh" (100), "update" (1), "thin" (1), and "iteration" (10000). It also has checkboxes for "over relax" and "adapting".

The status bar at the bottom indicates "updates took 6 s". The Windows taskbar at the very bottom shows the Start button and several open applications, including PeerSync, Internet Explorer, Lab3-Win..., Adobe Acrobat, WinBUGS14, RGui, WinEdt/Mi..., Yap 2.4.18..., and Microsoft P... The system clock shows 11:36 AM.

In the “node” entry type the name of the parameter for posterior inference, then click on “set”.

The screenshot displays the WinBUGS 14 software interface. The main window, titled "nonlinearregression", contains the following model specification:

```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha-beta*pow(gamma,x[i])
  }
  alpha ~ dnorm(0.0, 1.0E-6)
  beta ~ dnorm(0.0, 1.0E-6)
  gamma ~ dunif(0.0,1.0)
  tau ~ dgamma(0.01,0.01)
}
```

DATA

```
list(x=c(1.0,1.5,1.5,1.5,2.5,4.0,5.0,5.0,7.0,8.0,
8.5,9.0,9.5,9.5,10.0,12.0,12.0,13.0,13.0,14.5,
15.5,15.5,16.5,17.0,22.5,29.0,31.5),y=c(1.80,
1.85,1.87,1.77,2.02,2.27,2.15,2.26,2.47,2.19,
2.26,2.40,2.39,2.41,2.50,2.32,2.32,2.43,2.47,
2.56,2.65,2.47,2.64,2.56,2.70,2.72,2.57),N=27)
```

FIRST INITIAL VALUES  
`list(alpha=1,beta=1,tau=1,gamma=0.9)`

SECOND INITIAL VALUES ARE RANDOMLY DRAWN FROM THE PRIOR DISTRIBUTIONS.

Three tool windows are open:

- Specification Tool:** Contains buttons for "check model", "load data", "compile", "load inits", and "gen inits". It also has input fields for "num of chains" (set to 2) and "for chain" (set to 2).
- Update Tool:** Contains input fields for "updates" (10000), "refresh" (100), "update" (1), "thin" (1), and "iteration" (10000). It also has checkboxes for "over relax" and "adapting".
- Sample Monitor Tool:** Shows the selected node "alpha" with chains 1 to 2. It displays a table of percentiles: 2.5, 5, 10, 25, median, 75, 90, 95, 97.5. Buttons include "clear", "set", "trace", "history", "density", "stats", "coda", "quantiles", "lgr diag", and "auto cor".

The Windows taskbar at the bottom shows the Start button and several open applications, including PeerSync, Internet Explorer, Lab3-Win..., Adobe Acrobat, WinBUGS14, RGui, WinEdt, and Yap 2.4.18. The system clock shows 11:37 AM.



After all parameters have been “set”, type \* in the “node” window. This action will free all options in the “Sample Monitor Tool”, such as “trace”, “history”, “density”, “stats”, “quantiles” and “auto cor”.

The screenshot displays the WinBUGS 14 interface. The main window, titled "nonlinearregression", contains the following R code:

```

model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha-beta*pow(gamma,x[i])
  }
  alpha ~ dnorm(0.0, 1.0E-6)
  beta ~ dnorm(0.0, 1.0E-6)
  gamma ~ dunif(0.0,1.0)
  tau ~ dgamma(0.01,0.01)
}

```

DATA

```

list(x=c(1.0,1.5,1.5,1.5,2.5,4.0,5.0,5.0,7.0,8.0,
8.5,9.0,9.5,9.5,10.0,12.0,12.0,13.0,13.0,14.5,
15.5,15.5,16.5,17.0,22.5,29.0,31.5),y=c(1.80,
1.85,1.87,1.77,2.02,2.27,2.15,2.26,2.47,2.19,
2.26,2.40,2.39,2.41,2.50,2.32,2.32,2.43,2.47,
2.56,2.65,2.47,2.64,2.56,2.70,2.72,2.57),N=27)

```

FIRST INITIAL VALUES  
list(alpha=1,beta=1,tau=1,gamma=0.9)

SECOND INITIAL VALUES ARE RANDOMLY DRAWN FROM THE PRIOR DISTRIBUTIONS.

The interface also features several tool windows:

- Specification Tool:** Includes buttons for "check model", "load data", "compile", "load inits", and "gen inits". It also has a "num of chains" field set to 2 and a "for chain" field set to 2.
- Update Tool:** Includes fields for "updates" (10000), "refresh" (100), "update" (thin 1), and "iteration" (10000). It also has checkboxes for "over relax" and "adapting".
- Sample Monitor Tool:** A floating window with a "node" dropdown, "chains" 1 to 2, and a "percentiles" table. The table shows values for percentiles 2.5, 5, 10, 25, median, 75, 90, 95, and 97.5. Below the table are buttons for "clear", "set", "trace", "history", "density", "stats", "coda", "quantiles", "bgr diag", and "auto cor".

The Windows taskbar at the bottom shows the system tray with the time 11:39 AM and various application icons including PeerSync, Internet Explorer, Lab3-WinB..., Adobe Acr..., WinBUGS14, RGui, WinEdt/Mi..., Yap 2.4.18..., and Microsoft P...



“Update” for another 10000 draws and click on the option “trace” on the Sample Monitor Tool to see the evolution of the chains.

The screenshot displays the WinBUGS 14 interface with several windows open:

- nonlinearregression**: Contains the model code:

```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha-beta*pow(gamma,x[i])
  }
  alpha ~ dnorm(0,0.001)
  beta ~ dnorm(0,0.001)
  gamma ~ dunif(0,1)
  tau ~ dgamma(0.001,0.001)
}
```
- DATA**:

```
list(x=c(1.0,1.5,1.8,5,9,0,9,5,9,5,10,15.5,15.5,16.5,17,1.85,1.87,1.77,2.2,2.26,2.40,2.39,2.41,2.50,2.32,2.32,2.43,2.47,2.56,2.65,2.47,2.64,2.56,2.70,2.72,2.57),N=27)
```
- FIRST INITIAL VALUES**:

```
list(alpha=1,beta=1,tau=1,gamma=0.9)
```
- SECOND INITIAL VALUES ARE RANDOMLY DRAWN FROM THE PRIOR DISTRIBUTIONS.**
- Dynamic trace**: Shows four plots for alpha, beta, gamma, and tau chains over iterations 17650 to 17750.
- Specification Tool**: Includes buttons for 'check model', 'load data', 'compile', 'load inits', and 'gen inits'. It also shows 'num of chains' set to 2 and 'for chain' set to 2.
- Update Tool**: Shows 'updates' set to 10000, 'refresh' set to 100, 'update' set to thin 1, and 'iteration' set to 17900. It also has checkboxes for 'over relax' and 'adapting'.
- Sample Monitor Tool**: Shows 'node' as 'gamma', 'chains' 1 to 2, and a list of percentiles (2.5, 5, 10, 25, median, 75, 90, 95, 97.5).

# Click on the option “history” to see the history of the chains (trace plots).

The screenshot displays the WinBUGS 14 interface. The main window is titled "nonlinearregression" and contains the following model code:

```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha*beta*pow(gamma,x[i])
  }
  alpha ~ dnorm(0.0, 1.0E-6)
  beta ~ dnorm(0.0, 1.0E-6)
  gamma ~ dunif(0.0,1.0)
  tau ~ dgamma(0.01,0.01)
}
```

DATA

```
list(x=c(1.0,1.5,1.5,1.5,2.5,4.0,5.0,5.0,7.0,8.0,
8.5,9.0,9.5,9.5,10.0,12.0,12.0,13.0,13.0,14.5,
15.5,15.5,16.5,17.0,22.5,29.0,31.5),y=c(1.80,
1.85,1.87,1.77,2.02,2.27,2.15,2.26,2.47,2.19,
2.26,2.40,2.39,2.41,2.50,2.32,2.32,2.43,2.47,
2.56,2.65,2.47,2.64,2.56,2.70,2.72,2.57),N=27)
```

FIRST INITIAL VALUES

```
list(alpha=1,beta=1,tau=1,gamma=0.9)
```

SECOND INITIAL VALUES ARE RANDOMLY DRAWN FROM THE PRIOR DISTRIBUTIONS.

Three tool windows are open:

- Specification Tool:** Includes buttons for "check model", "load data", "complete", "load inits", "gen inits", and "for chain" set to 2.
- Update Tool:** Shows "updates" as 10000, "refresh" as 100, "update" as "thin 1", and "iteration" as 20000. Checkboxes for "over relax" and "adapting" are present.
- Sample Monitor Tool:** Shows "node" as "alpha", "chains" as 1 to 2, "percentiles" as 2.5, 5, 10, 25, 75, 90, 95, 97.5, "beg" as 1, "end" as 1000000, and "thin" as 1. Buttons for "clear", "set", "trace", "history", "density", "stats", "coda", "quantiles", "bgr diag", and "auto cor" are visible.

The "Time series" window on the right displays four trace plots for parameters alpha, beta, gamma, and tau, each showing two chains (1 and 2) over 20,000 iterations. The y-axis for alpha ranges from 2.0 to 4.0, beta from 0.5 to 2.0, gamma from 0.4 to 1.0, and tau from 0.0 to 300.0.

Try other options, such as “density”, “auto cor” and “stats”. For instance, the posterior mean of alpha is 2.65, while the posterior variance of alpha is 0.07826. The Monte Carlo error is 0.002619 when computing these two moments of alpha. The statistical summary is based on the MCMC chain starting at draw 10001 and finishing at draw 20000, i.e., a total of 10000 draws.

The screenshot displays the WinBUGS 14 interface for a nonlinear regression model. The main window shows the model code, data, and initial values. Three diagnostic plots are open: Kernel density, Autocorrelation function, and Node statistics.

**Model Code:**

```

model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha-beta*pow(gamma,x[i])
  }
  alpha ~ dnorm(0.0, 1.0E-6)
  beta ~ dnorm(0.0, 1.0E-6)
  gamma ~ dunif(0.0,1.0)
  tau ~ dgamma(0.01,0.01)
}

```

**Data:**

```

list(x=c(1.0,1.5,1.5,1.5,2.5,4.0,5.0,5.0,7.0,8.0,
8.5,9.0,9.5,9.5,10.0,12.0,12.0,13.0,13.0,14.5,
15.5,15.5,16.5,17.0,22.5,29.0,31.5),y=c(1.80,
1.85,1.87,1.77,2.02,2.27,2.15,2.26,2.47,2.19,
2.26,2.40,2.39,2.41,2.50,2.32,2.32,2.43,2.47,
2.56,2.65,2.47,2.64,2.56,2.70,2.72,2.57),N=27)

```

**First Initial Values:**

```
list(alpha=1,beta=1,tau=1,gamma=0.9)
```

**Second Initial Values:** ARE RANDOMLY DRAWN FROM THE PRIOR DISTRIBUTIONS.

**Kernel density plots:** Four plots showing the posterior distributions for alpha, beta, gamma, and tau. Each plot is titled "chains 1:2 sample: 20000".

**Autocorrelation function plots:** Four plots showing the autocorrelation function for alpha, beta, gamma, and tau. Each plot is titled "chains 1:2".

**Node statistics table:**

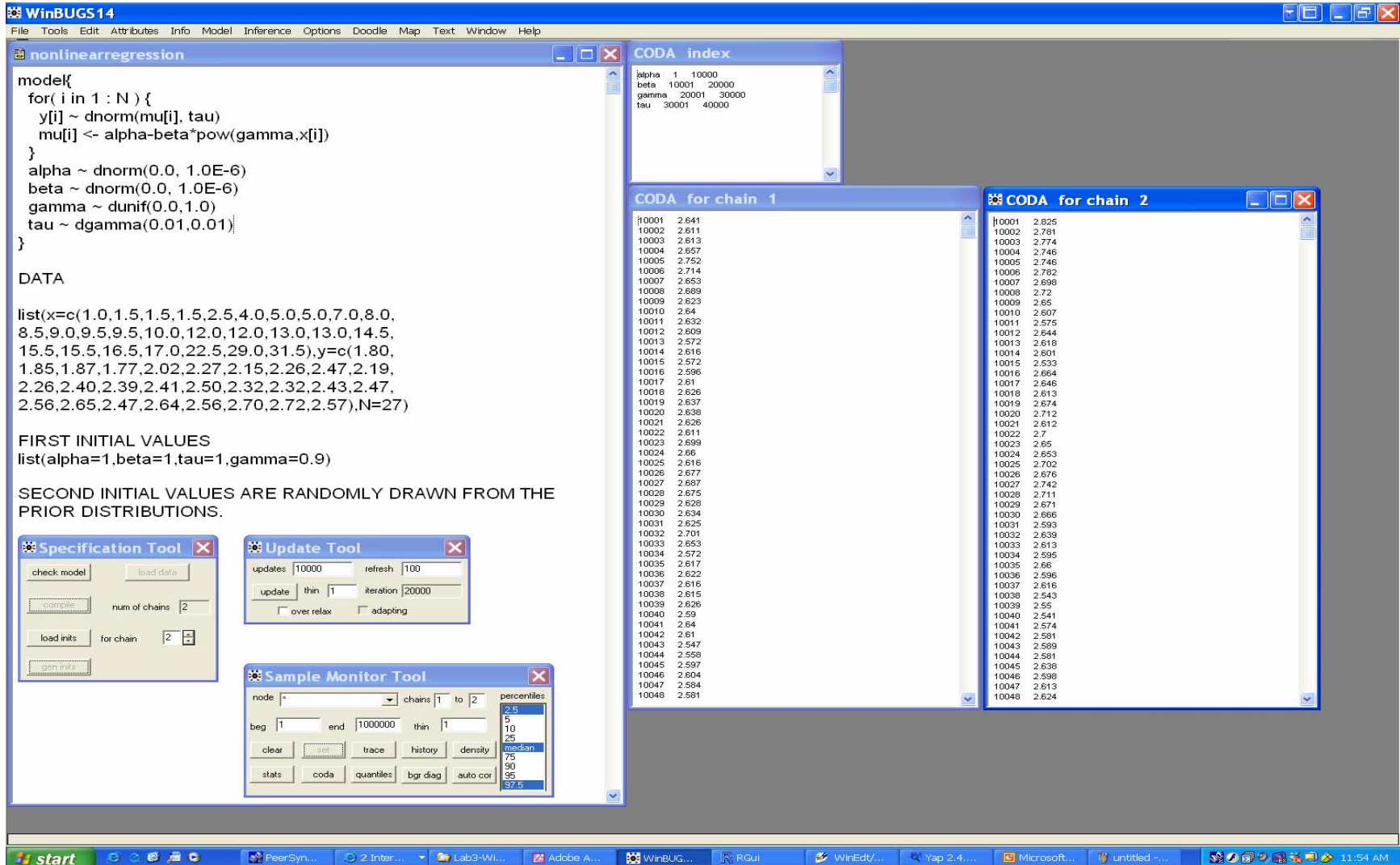
node	mean	sd	MC error	2.5%	median	97.5%	start	sample
alpha	2.651	0.07826	0.002619	2.524	2.644	2.819	10001	20000
beta	0.9751	0.08147	9.803E-4	0.8217	0.9735	1.139	10001	20000
gamma	0.8608	0.03497	0.001163	0.7814	0.8647	0.9174	10001	20000
tau	100.8	29.48	0.2606	51.56	97.89	167.2	10001	20000

**Specification Tool:** Includes buttons for "check model", "load data", "complete", "load inits", and "gen inits".

**Update Tool:** Includes fields for "updates" (10000), "refresh" (100), "update" (thin 1), and "iteration" (20000). Includes checkboxes for "over relax" and "adapting".

**Sample Monitor Tool:** Includes fields for "node", "chains" (1 to 2), "percentiles" (2.5, 5, 10, 25, median, 75, 90, 95, 97.5), "beg" (1), "end" (1000000), and "thin" (1). Includes buttons for "clear", "set", "trace", "history", "density", "stats", "coda", "quantiles", "bgr diag", and "auto cor".

Save the output of your MCMC by clicking on “coda”. Three files pop up: CODA index, CODA for chain 1 and CODA for chain 2. The file CODA index tells you how to read the other two files. Now you are ready to upload these two chains in your favorite statistical package (R, Matlab, etc) and create your own statistical summaries, plots, etc.

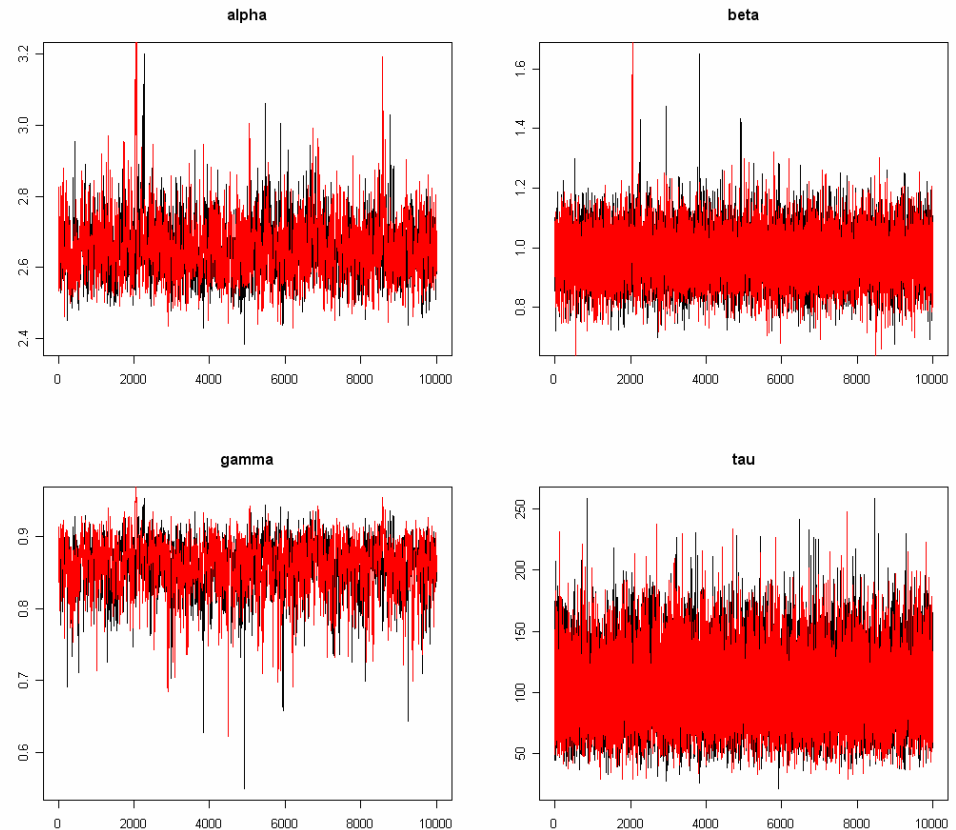


# Suppose you save the two chains in files chain1.txt and chain2.txt

```
#####
# DATA
#####
x = c(1.0,1.5,1.5,1.5,2.5,4.0,5.0,5.0,7.0,8.0,8.5,9.0,9.5,9.5,10.0,12.0,
12.0,13.0,13.0,14.5,15.5,15.5,16.5,17.0,22.5,29.0,31.5)
y = c(1.80,1.85,1.87,1.77,2.02,2.27,2.15,2.26,2.47,2.19,2.26,2.40,2.39,
2.41,2.50,2.32,2.32,2.43, 2.47,2.56,2.65,2.47,2.64,2.56,2.70,2.72,2.57)
plot(x,y)

#####
# Reading WINBUGS output (2 MCMC chains of length 10000)
#####
M = 10000
chain1 = matrix(scan("chain1.txt"),4*M,2,byrow=T)
chain2 = matrix(scan("chain2.txt"),4*M,2,byrow=T)
chain1 = matrix(chain1[,2],M,4)
chain2 = matrix(chain2[,2],M,4)

names = c("alpha","beta","gamma","tau")
par(mfrow=c(2,2))
for (i in 1:4){
  ts.plot(chain1[,i],xlab="",ylab="",main="")
  title(names[i])
  lines(chain2[,i],col=2)
}
```



# Posterior distribution of the nonlinear mean function $\mu(x) = \alpha - \beta\gamma^x$

```
# quantile functions
quantile025 = function(x){quantile(x,0.025)}
quantile975 = function(x){quantile(x,0.975)}

# one larger chain with 20000 draws
chain = rbind(chain1,chain2)
M = nrow(chain)

# computing mu(x) for several values of x
xs = 1:32
meanfunction = matrix(0,M,32)
for (i in 1:32)
  meanfunction[,i] = chain[,1] - chain[,2]*chain[,3]^xs[i]
meanf = apply(meanfunction,2,mean)
q025 = apply(meanfunction,2,quantile025)
q975 = apply(meanfunction,2,quantile975)
plot(x,y)
lines(xs,meanf,col=2)
lines(xs,q025,col=4,lty=2)
lines(xs,q975,col=4,lty=2)
```

