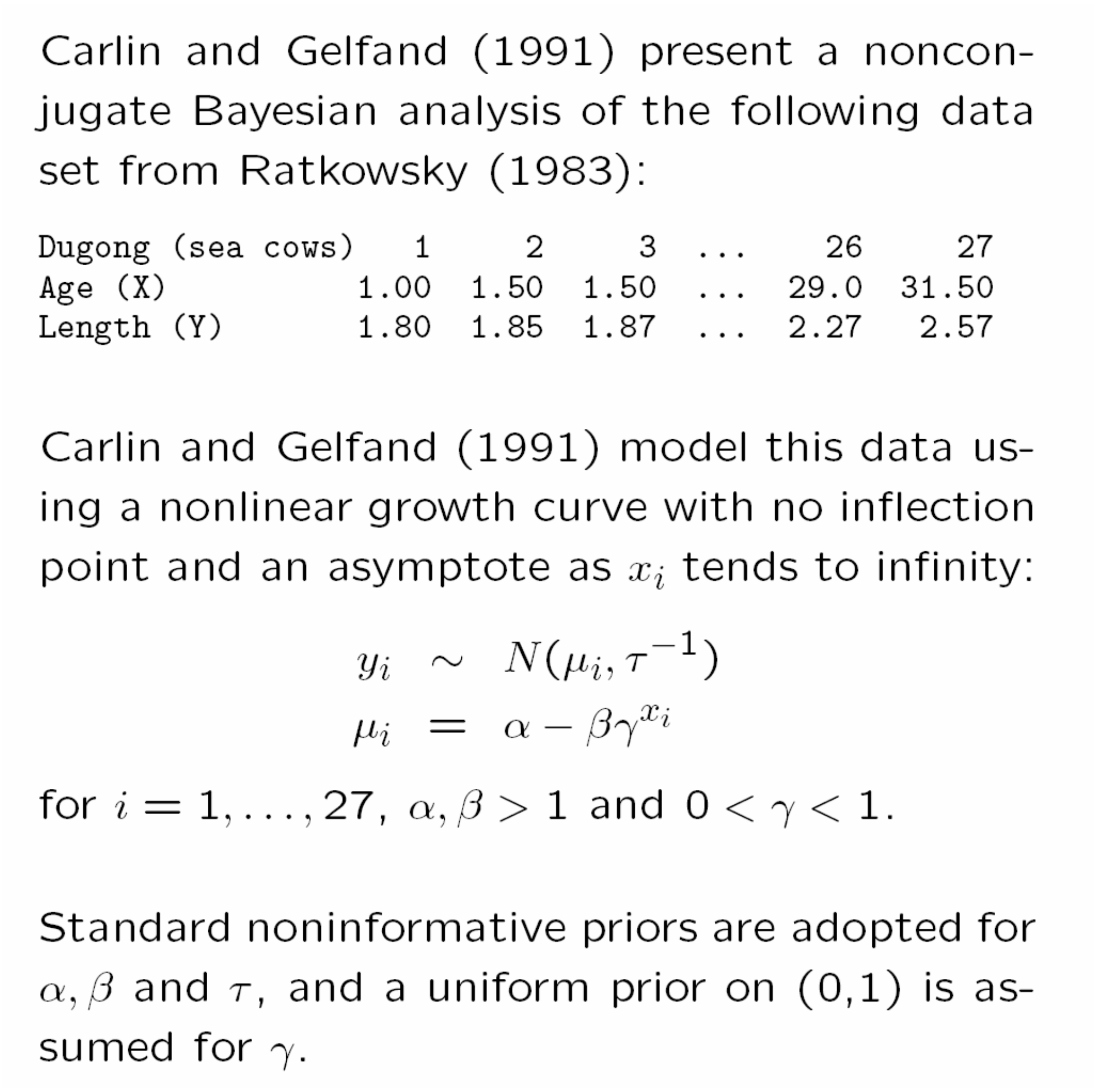
Brief WinBUGS tutorial

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:

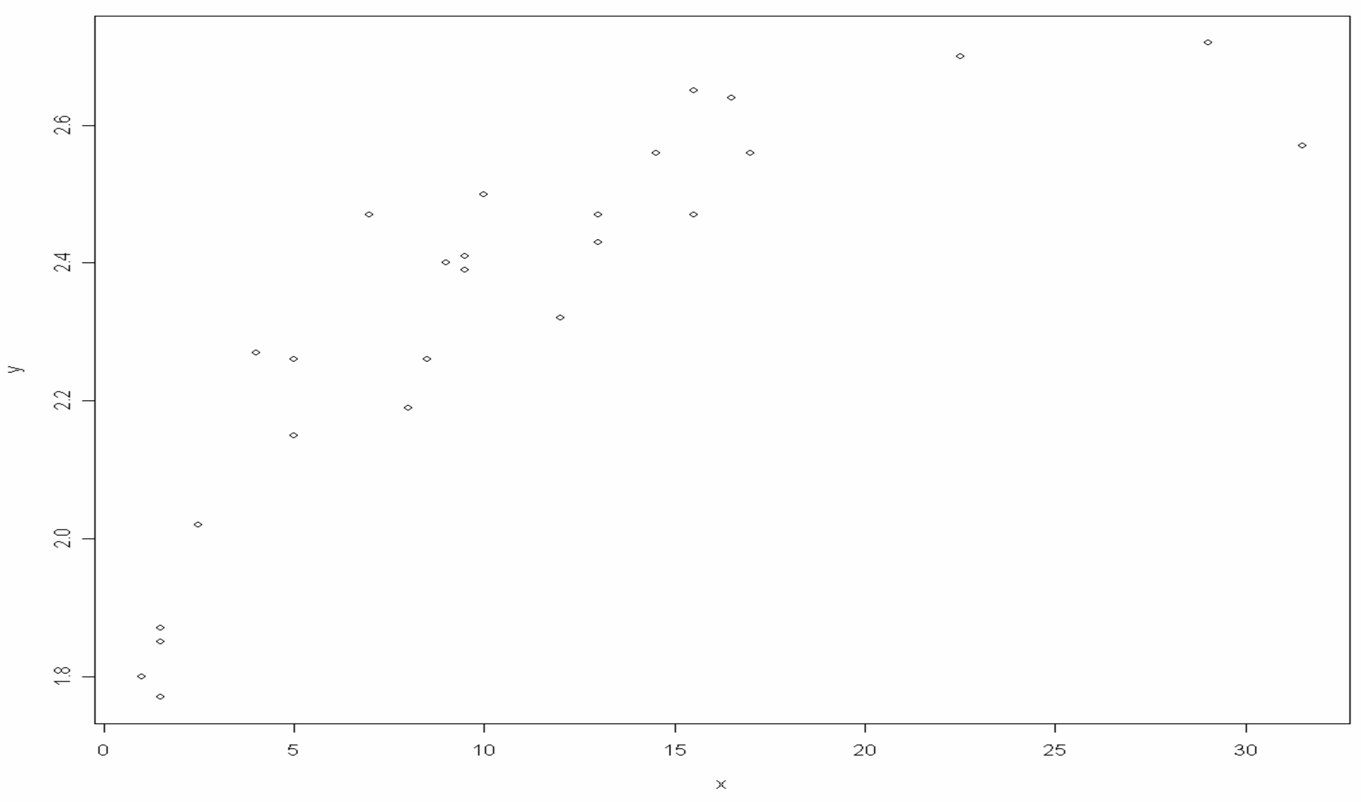
$$\begin{aligned} y_i &\sim N(\mu_i, \tau^{-1}) \\ \mu_i &= \alpha - \beta\gamma^{x_i} \end{aligned}$$

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

Standard noninformative priors are adopted for α, β and τ, and a uniform prior on (0,1) is assumed for γ.

Example from the winbugs manual at http://www.mrc-bsu.cam.ac.uk/bugs/

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



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

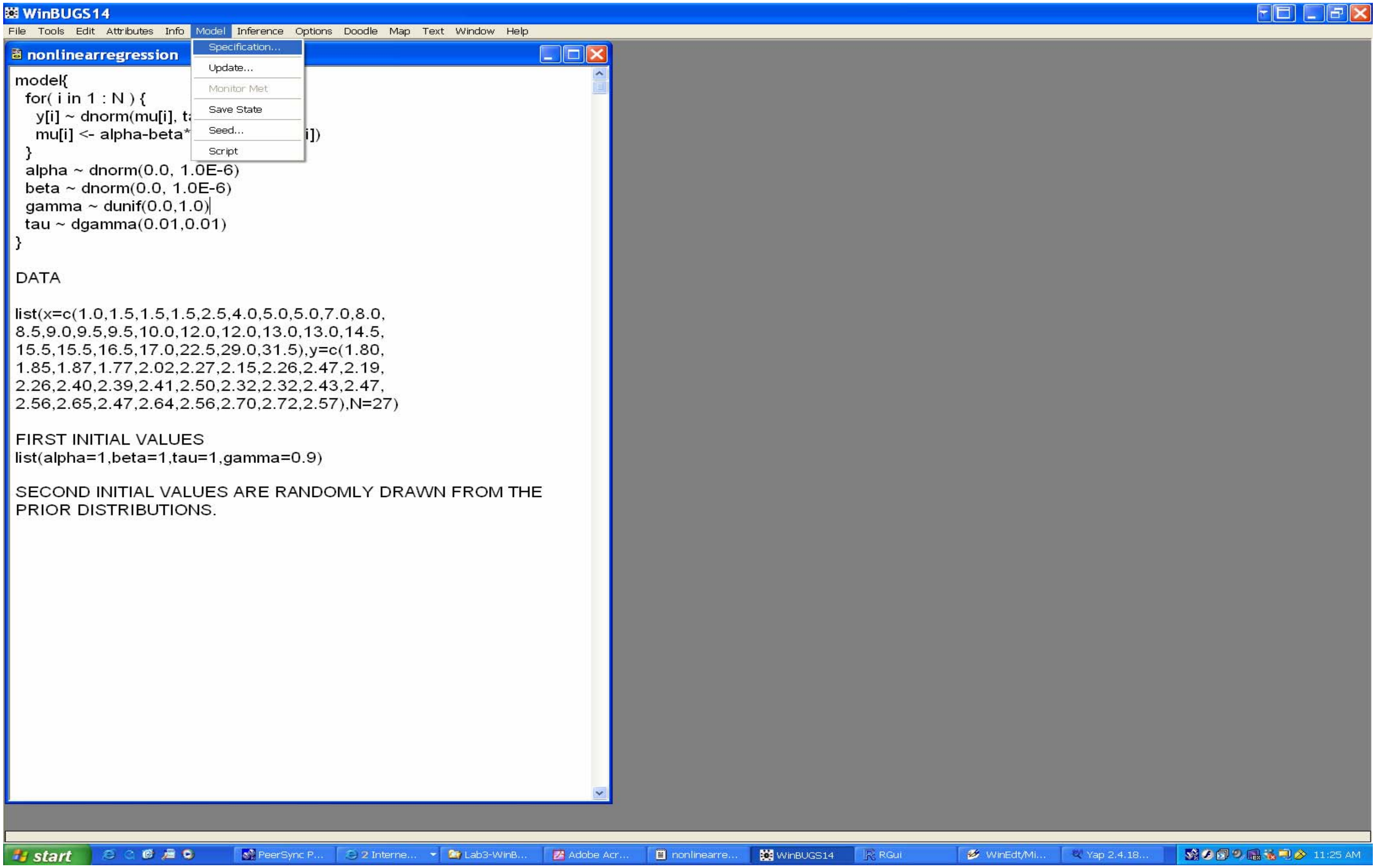
```
list(x=c(1.0,1.5,1.5,1.5,2.5,4.0,5.0,5.0,7.0,8.0,
8.5,9.0,9.5,9.5,10.0,12.0,12.0,13.0,13.0,14.5,
15.5,15.5,16.5,17.0,22.5,29.0,31.5),y=c(1.80,
1.85,1.87,1.77,2.02,2.27,2.15,2.26,2.47,2.19,
2.26,2.40,2.39,2.41,2.50,2.32,2.32,2.43,2.47,
2.56,2.65,2.47,2.64,2.56,2.70,2.72,2.57),N=27)
```

FIRST INITIAL VALUES

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

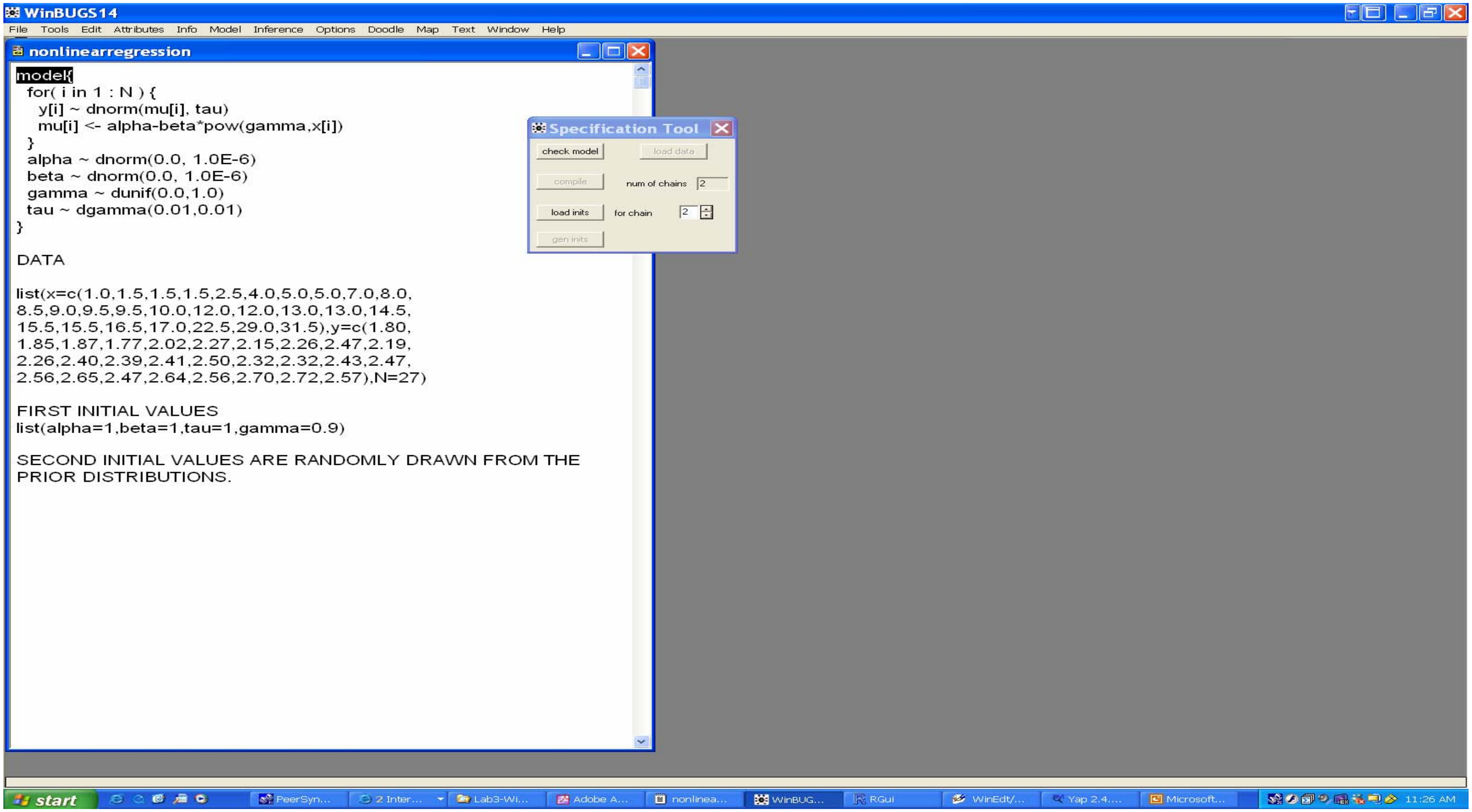
SECOND INITIAL VALUES ARE RANDOMLY DRAWN FROM THE PRIOR DISTRIBUTIONS.

Model>Specification



```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], t
    mu[i] <- alpha-beta*                    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.
```

Select model, then “check model”



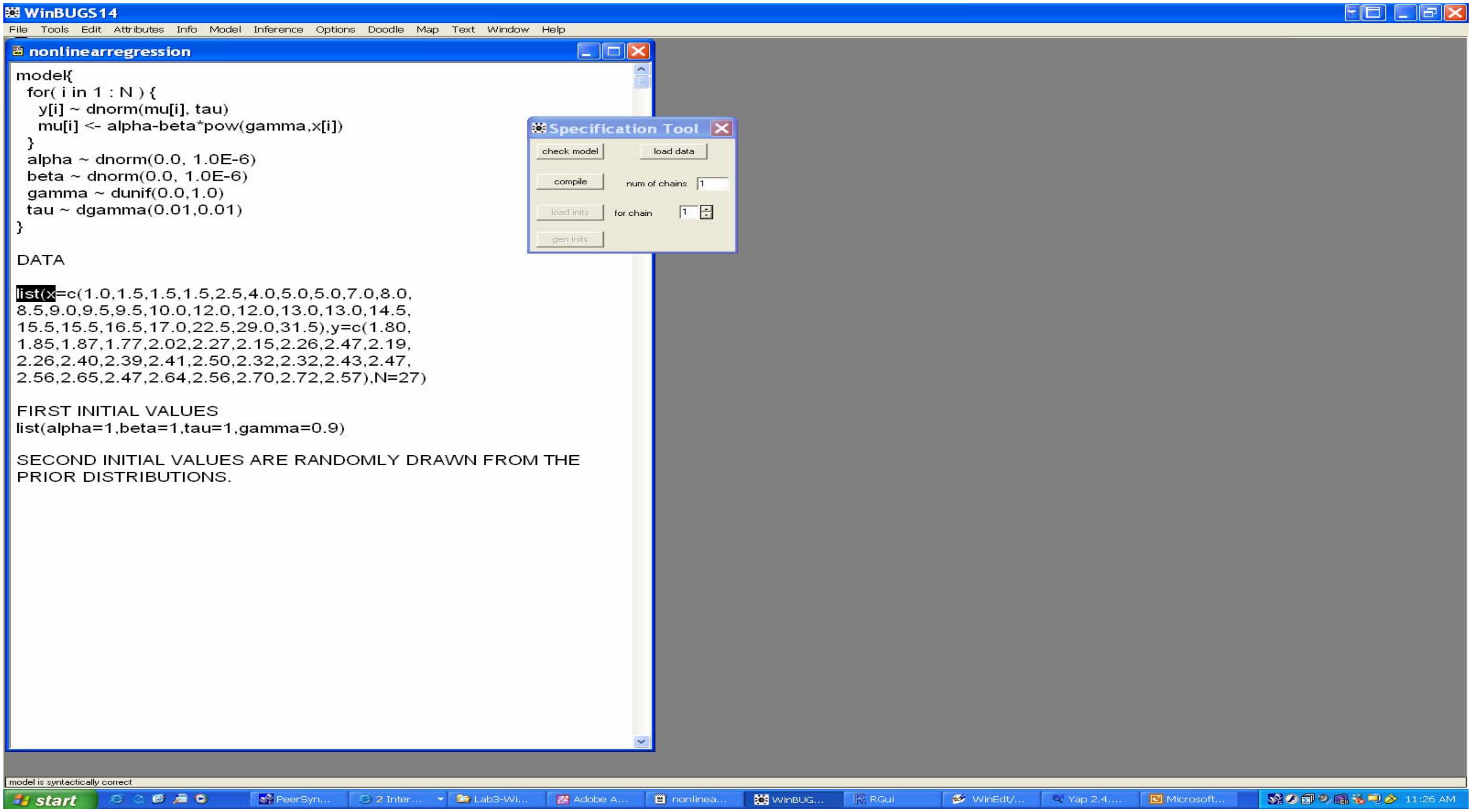
```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha-beta*pow(gamma,x[i])
  }
  alpha ~ dnorm(0.0, 1.0E-6)
  beta ~ dnorm(0.0, 1.0E-6)
  gamma ~ dunif(0.0,1.0)
  tau ~ dgamma(0.01,0.01)
}

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.
```

num of chains 2

for chain 2

Bottom message: “model is syntactically correct”

Select data, then “load data”

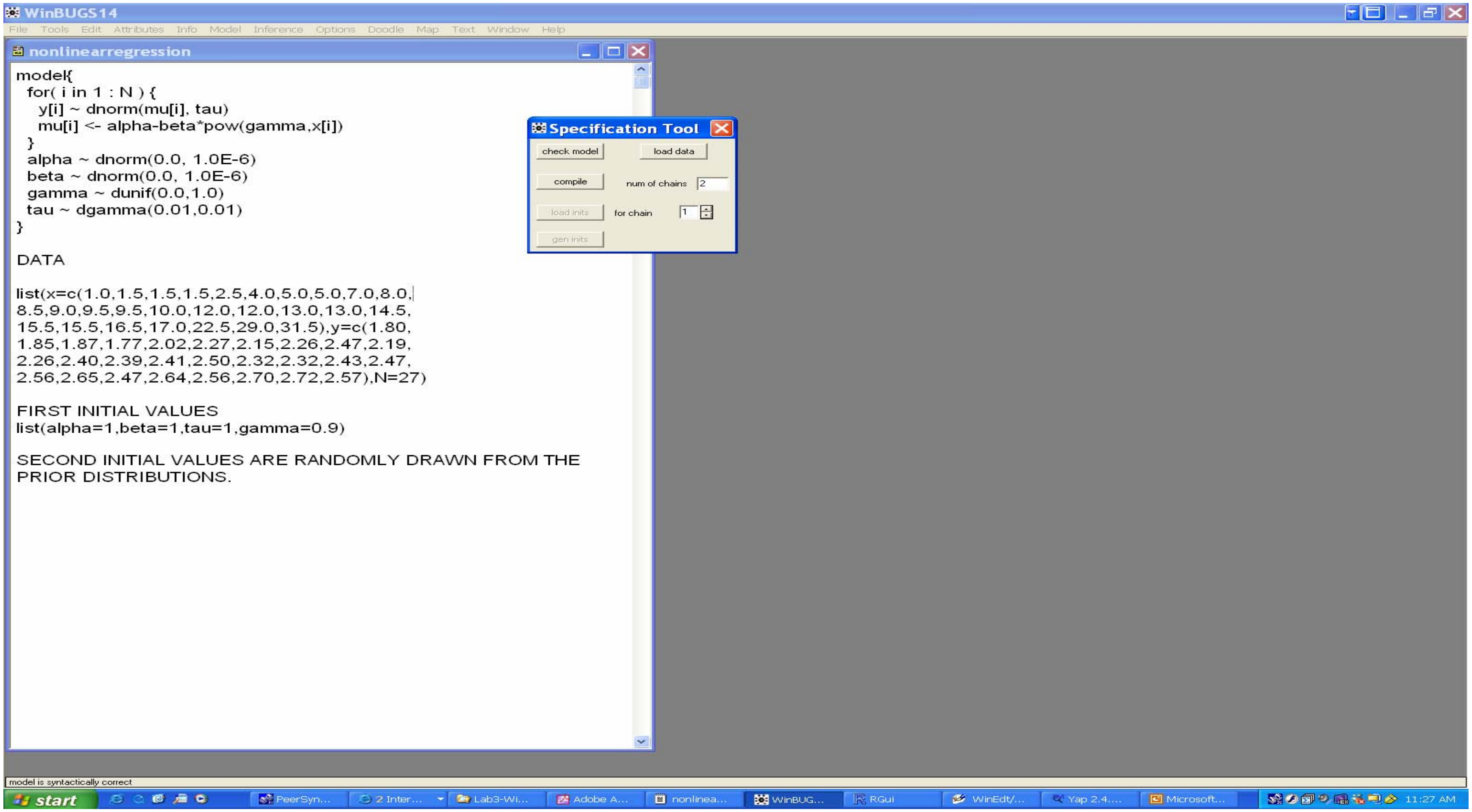


```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha-beta*pow(gamma,x[i])
  }
  alpha ~ dnorm(0.0, 1.0E-6)
  beta ~ dnorm(0.0, 1.0E-6)
  gamma ~ dunif(0.0,1.0)
  tau ~ dgamma(0.01,0.01)
}

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.
```

model is syntactically correct

Bottom message: “data loaded”

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



```
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.
```

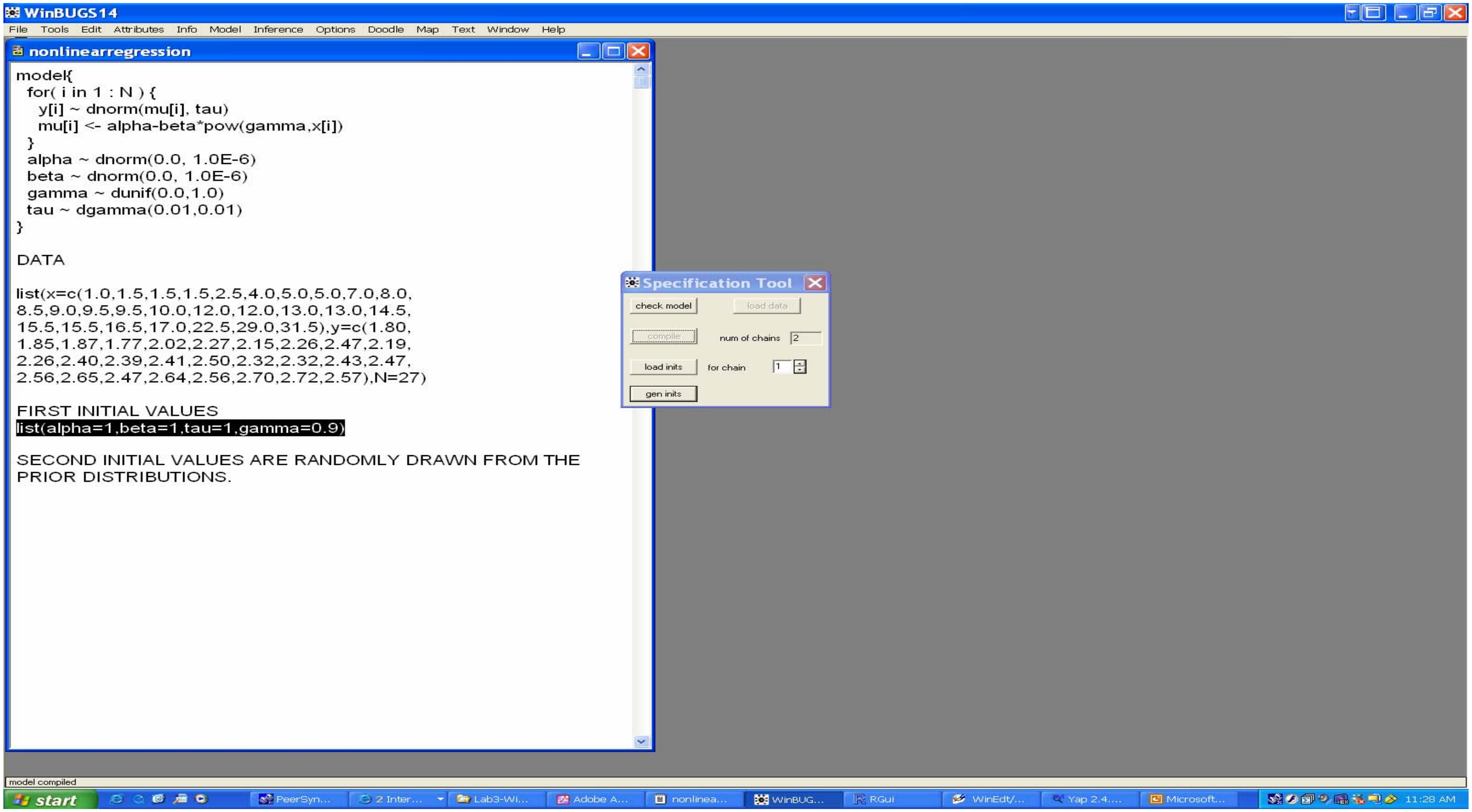
num of chains 2

for chain 1

model is syntactically correct

Bottom message: “model compiled”.

“load inits” for the 1st chain.



```
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.
```

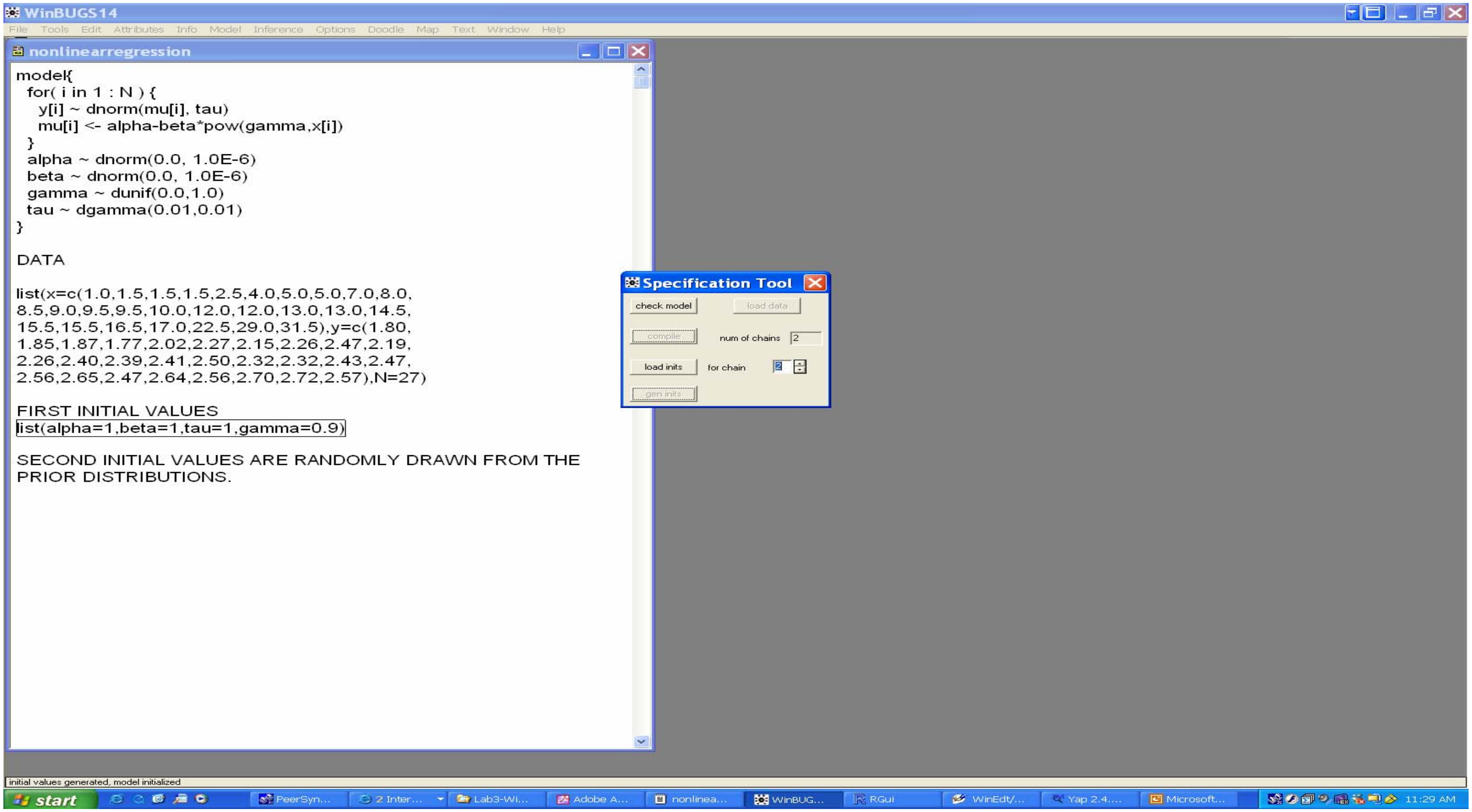
num of chains 2

for chain 1

model compiled

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

“gen inits” for the 2nd chain.



```
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.
```

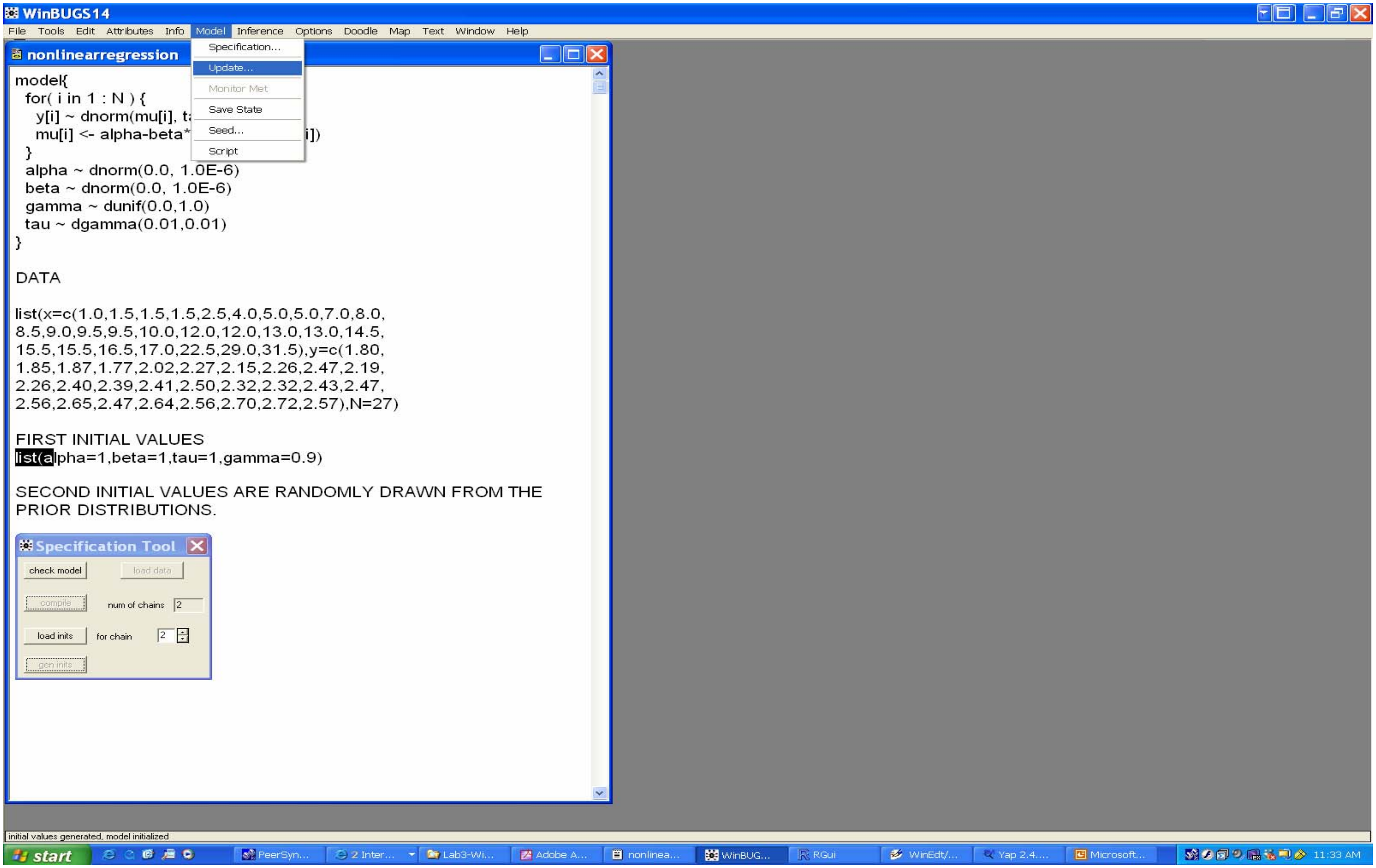
num of chains 2

for chain 2

initial values generated, model initialized

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

Model > Update



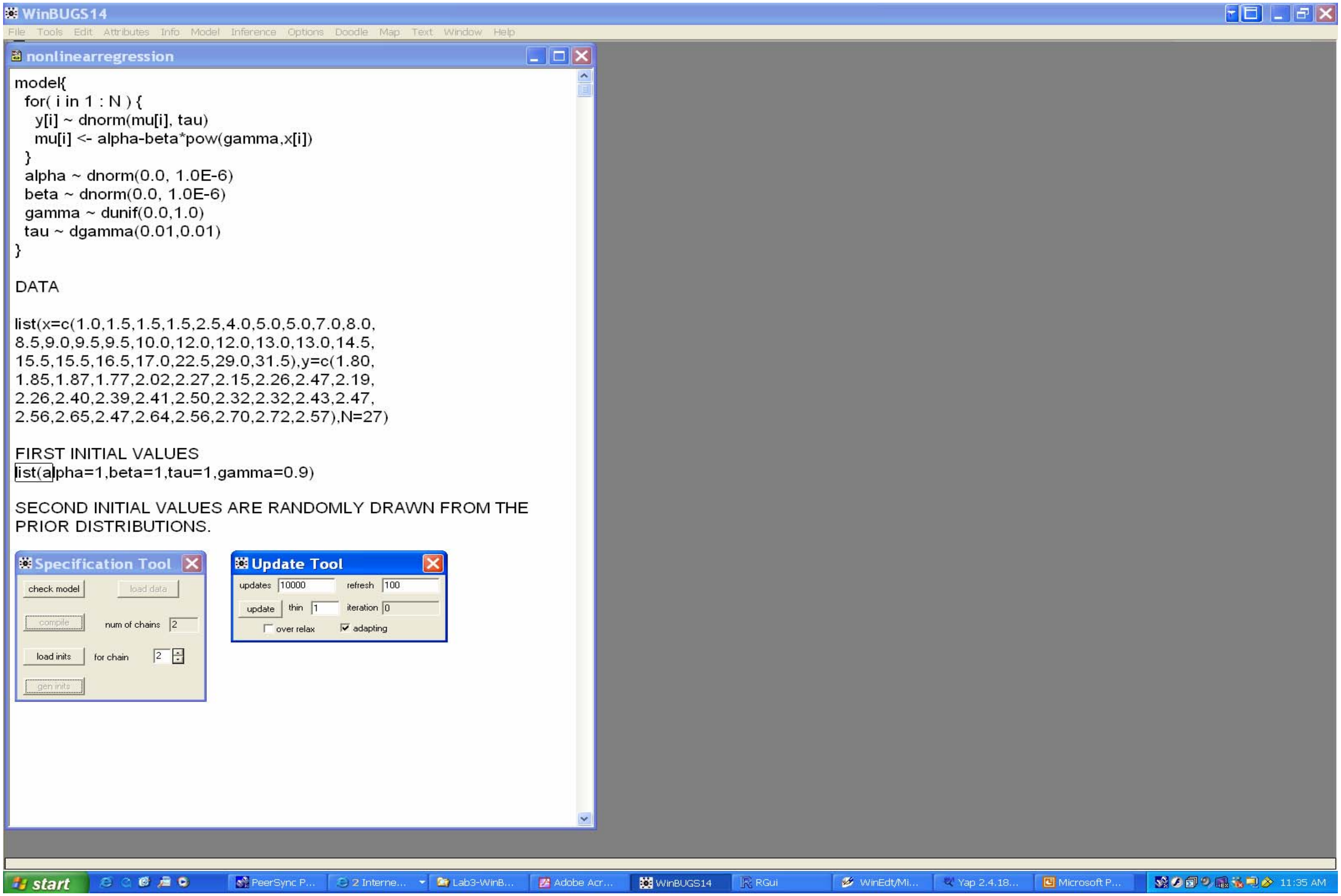
```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], t
    mu[i] <- alpha-beta*                    i])
  }
  alpha ~ dnorm(0.0, 1.0E-6)
  beta ~ dnorm(0.0, 1.0E-6)
  gamma ~ dunif(0.0,1.0)
  tau ~ dgamma(0.01,0.01)
}

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.
```

num of chains 2

for chain 2

initial values generated, model initialized

Let us run the MCMC for 10000 iterations



```
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

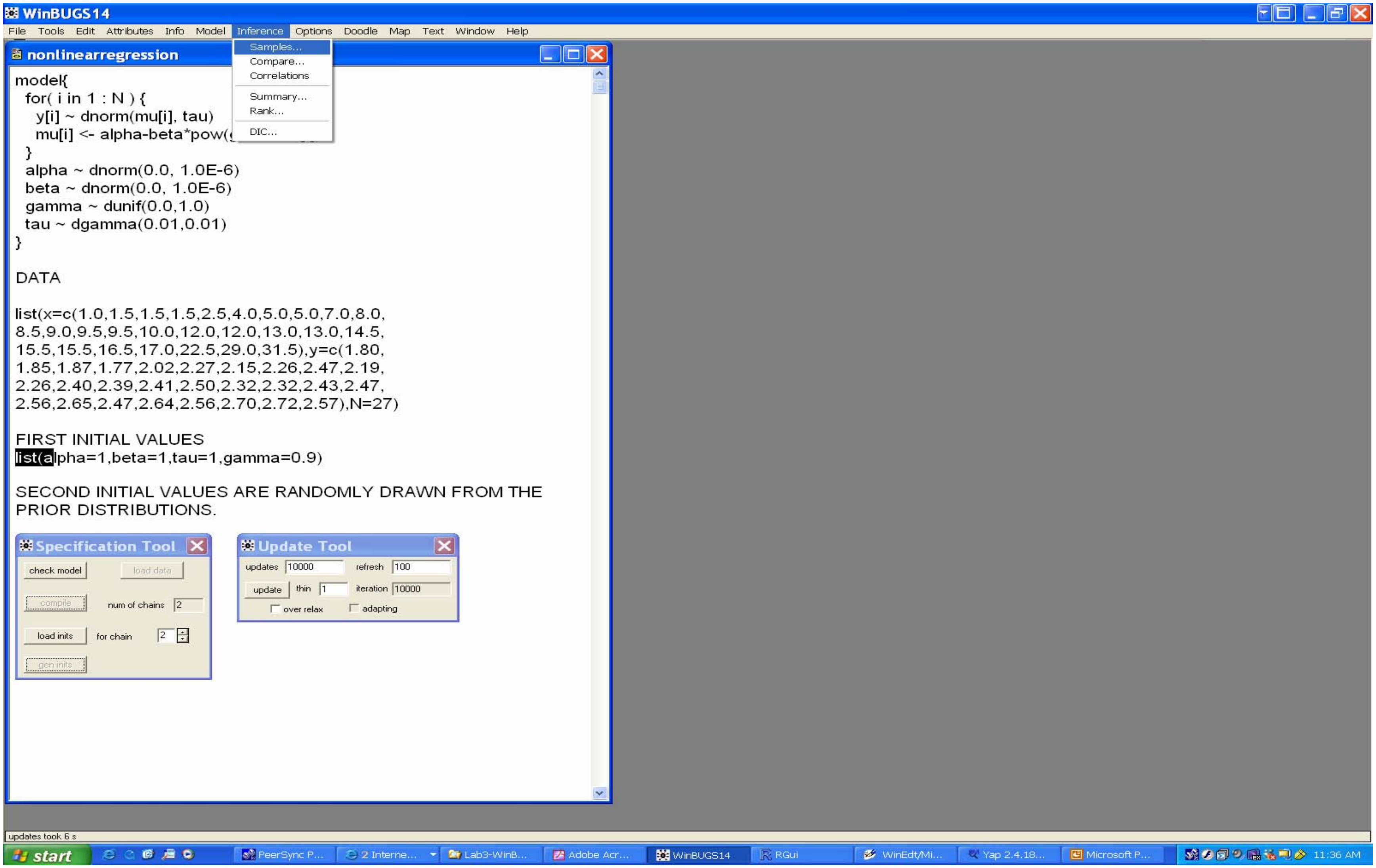
```
list(alpha=1,beta=1,tau=1,gamma=0.9)
```

SECOND INITIAL VALUES ARE RANDOMLY DRAWN FROM THE PRIOR DISTRIBUTIONS.

Specification Tool: num of chains 2; for chain 2

Update Tool: updates 10000; refresh 100; thin 1; iteration 0; ☐ over relax; ☑ adapting

Inference>sample



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

num of chains 2

for chain 2

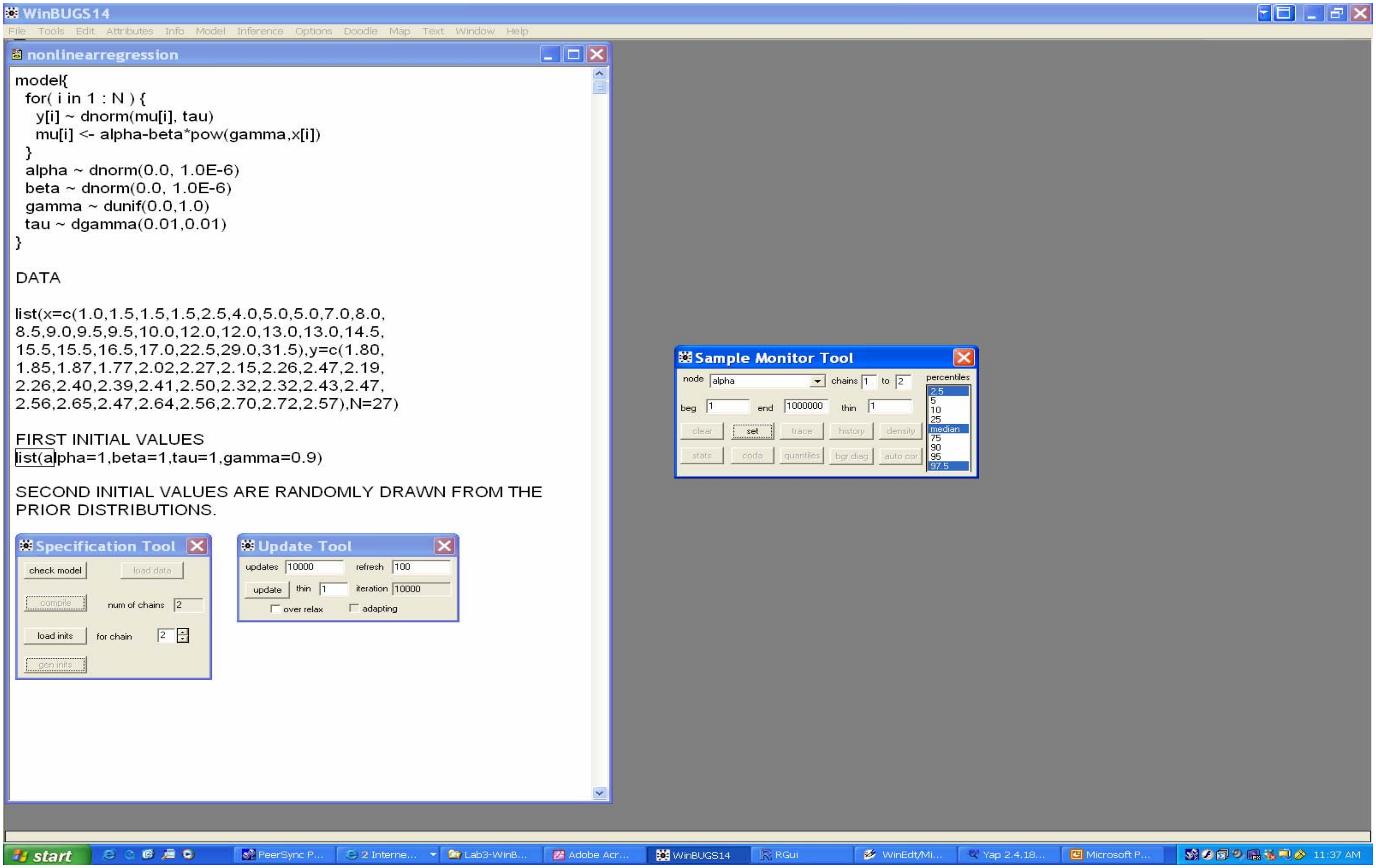
updates 10000 refresh 100

thin 1 iteration 10000

□ over relax □ adapting

updates took 6 s

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



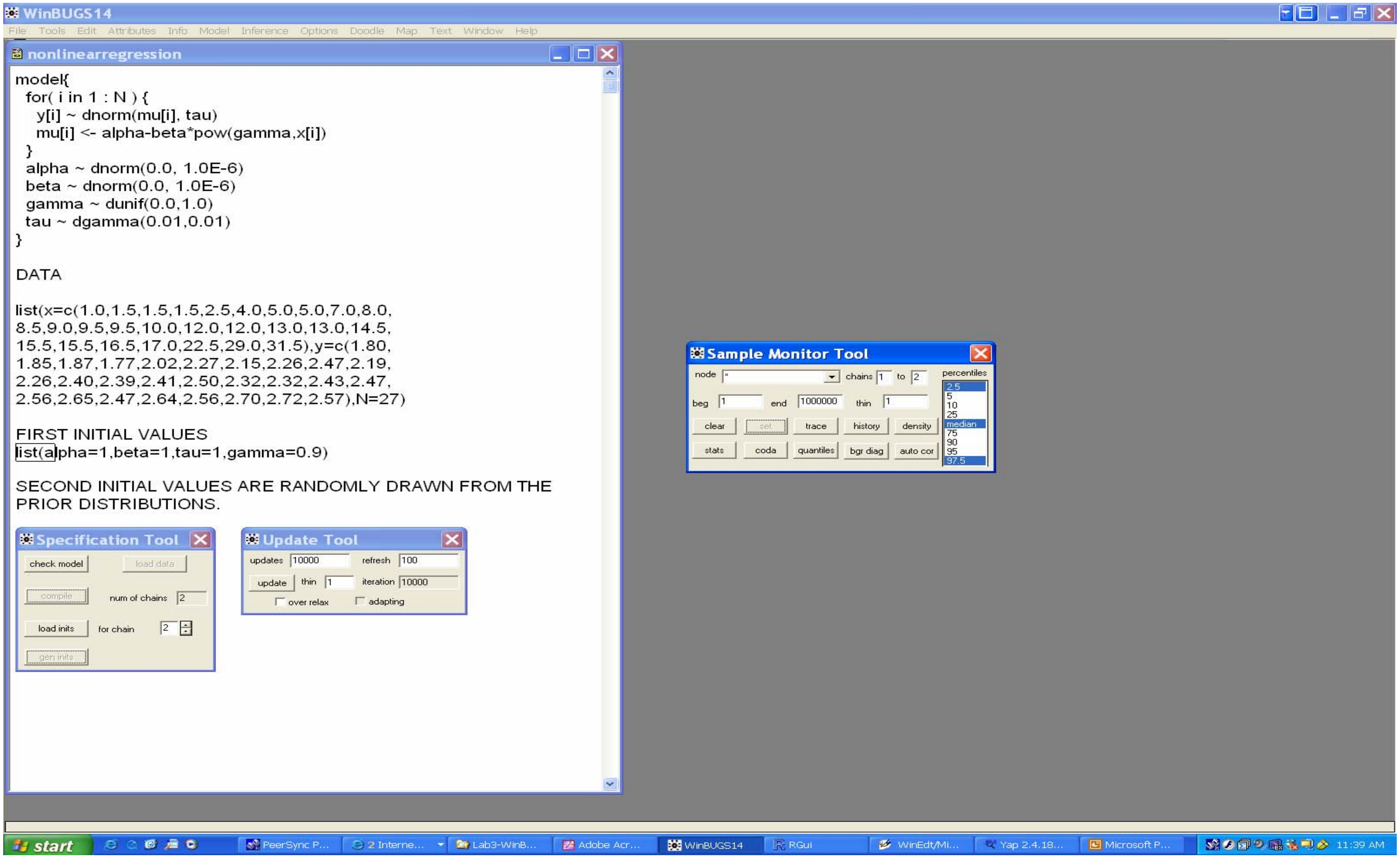
```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha-beta*pow(gamma,x[i])
  }
  alpha ~ dnorm(0.0, 1.0E-6)
  beta ~ dnorm(0.0, 1.0E-6)
  gamma ~ dunif(0.0,1.0)
  tau ~ dgamma(0.01,0.01)
}

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.
```

Specification Tool: num of chains 2; for chain 2

Update Tool: updates 10000; refresh 100; thin 1; iteration 10000; □ over relax; □ adapting

Sample Monitor Tool: node alpha; chains 1 to 2; beg 1; end 1000000; thin 1; percentiles 2.5, 5, 10, 25, median, 75, 90, 95, 97.5

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".



```
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.
```

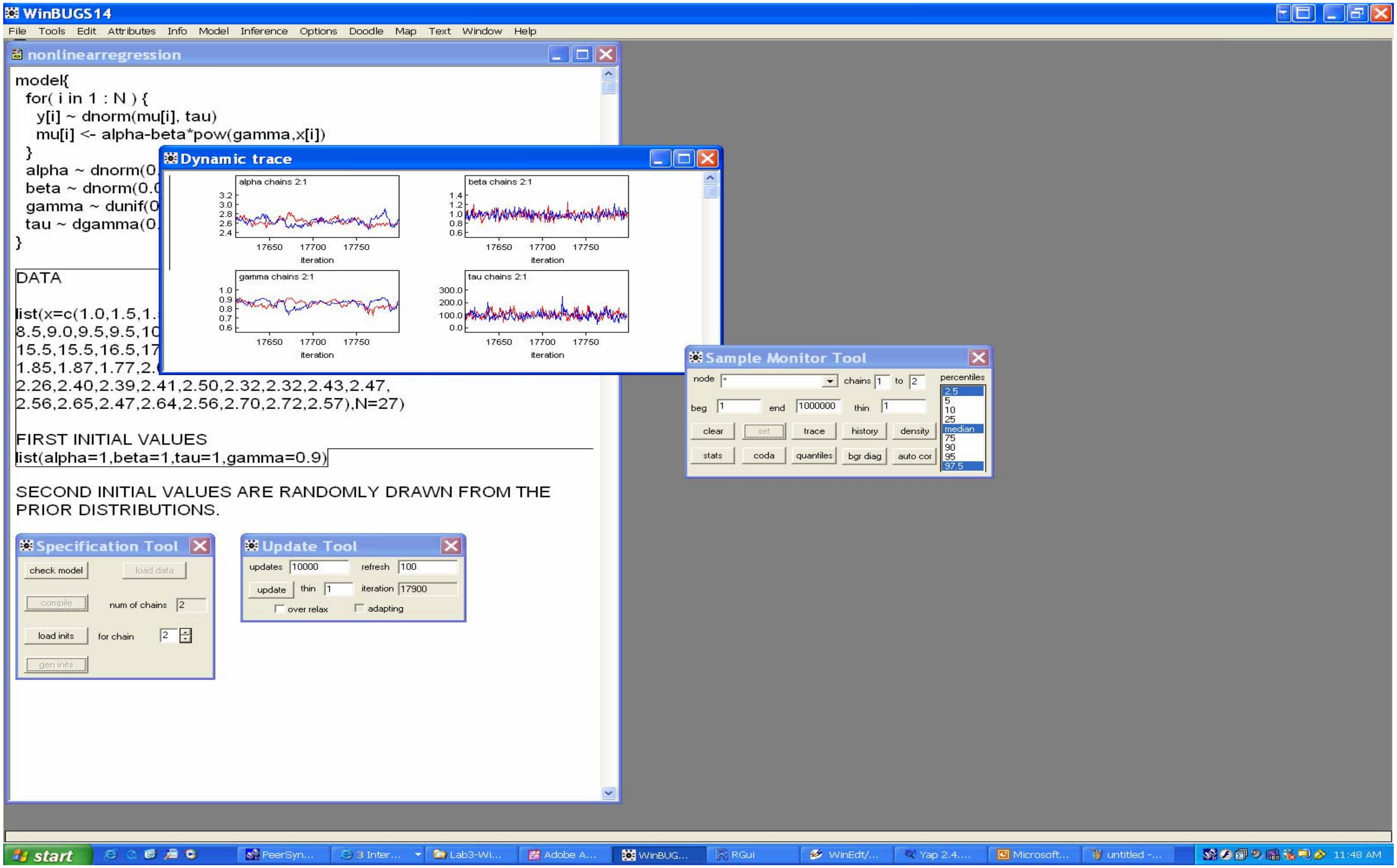
Specification Tool: num of chains 2; for chain 2

Update Tool: updates 10000; refresh 100; thin 1; iteration 10000; □ over relax; □ adapting

Sample Monitor Tool: node *; chains 1 to 2; beg 1; end 1000000; thin 1

percentiles: 2.5 (selected), 5, 10, 25, median (selected), 75, 90, 95, 97.5 (selected)

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



```
model{
  for( i in 1 : N ) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha-beta*pow(gamma,x[i])
  }
  alpha ~ dnorm(0.
  beta ~ dnorm(0.0
  gamma ~ dunif(0
  tau ~ dgamma(0.
}
```

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.
2.26,2.40,2.39,2.41,2.50,2.32,2.32,2.43,2.47,
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)
```

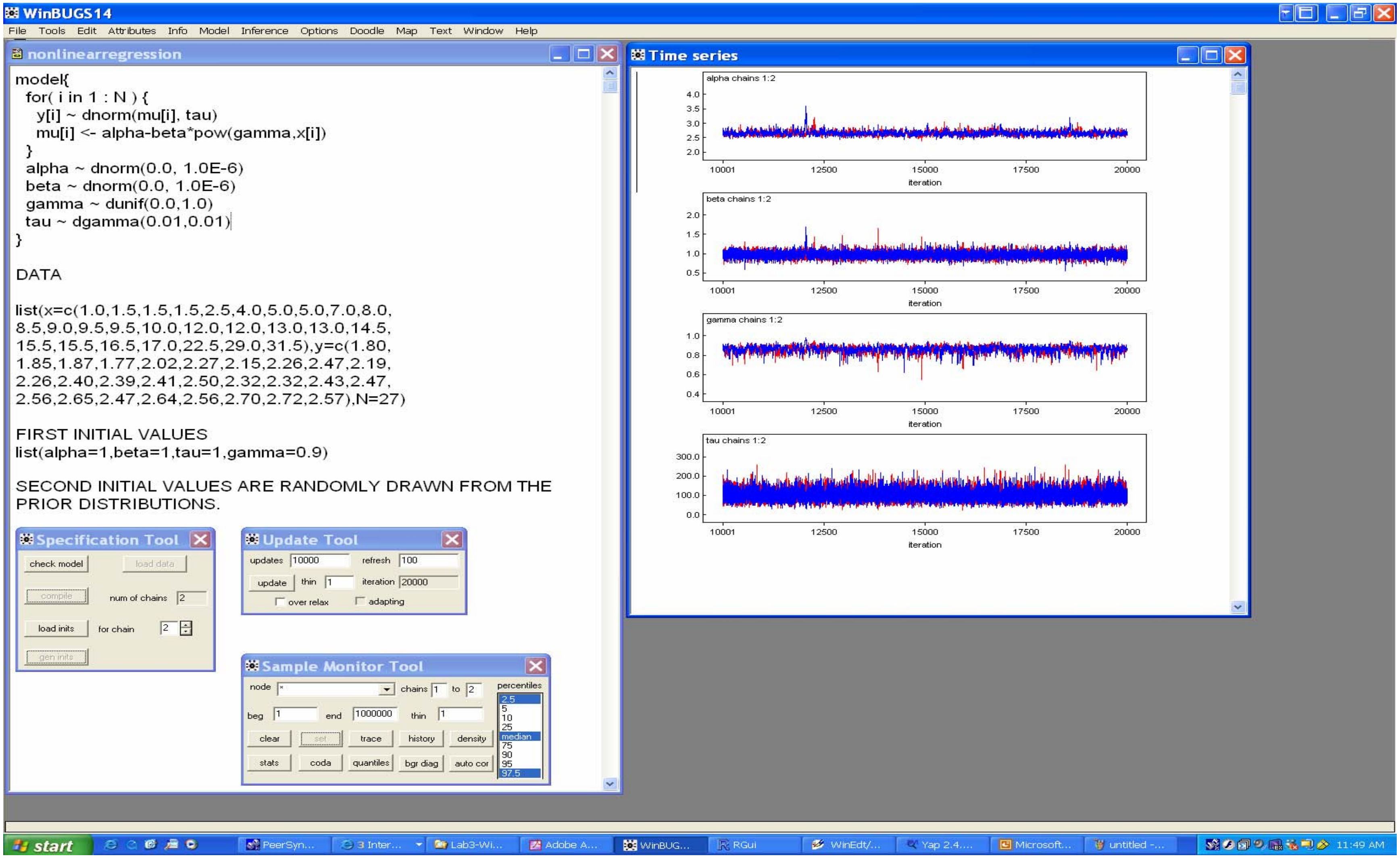
SECOND INITIAL VALUES ARE RANDOMLY DRAWN FROM THE PRIOR DISTRIBUTIONS.

Sample Monitor Tool: node * chains 1 to 2; beg 1 end 1000000 thin 1; percentiles 2.5 5 10 25 median 75 90 95 97.5

Specification Tool: num of chains 2; for chain 2

Update Tool: updates 10000 refresh 100; thin 1 iteration 17900; □ over relax □ adapting

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



WinBUGS14

nonlinearregression

```
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.

Specification Tool

num of chains: 2

for chain: 2

Update Tool

updates: 10000 refresh: 100

thin: 1 iteration: 20000

□ over relax □ adapting

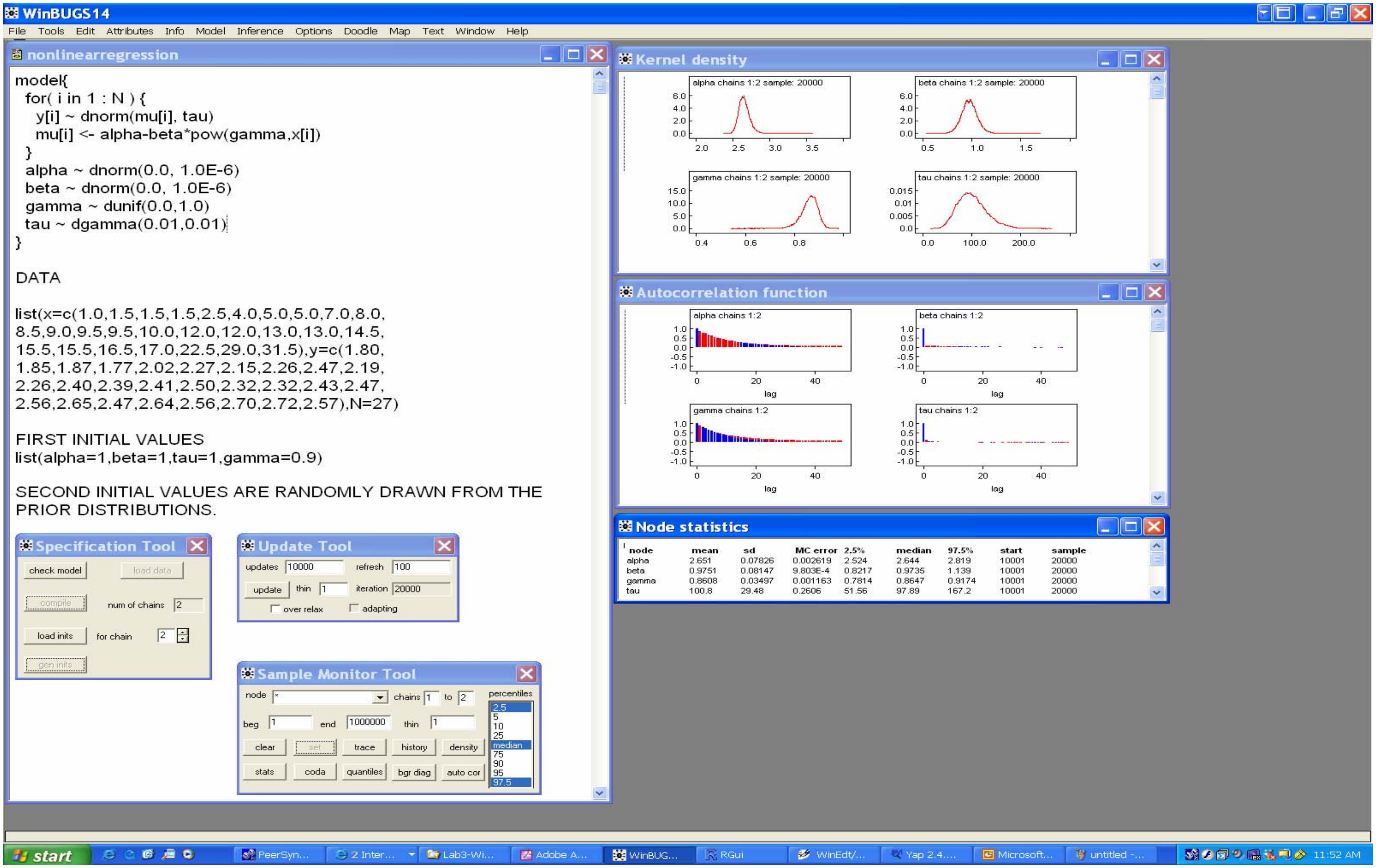
Sample Monitor Tool

node: * chains: 1 to 2

beg: 1 end: 1000000 thin: 1

Time series

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.



```
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.
```

num of chains 2

for chain 2

updates 10000 refresh 100

thin 1 iteration 20000

□ over relax □ adapting

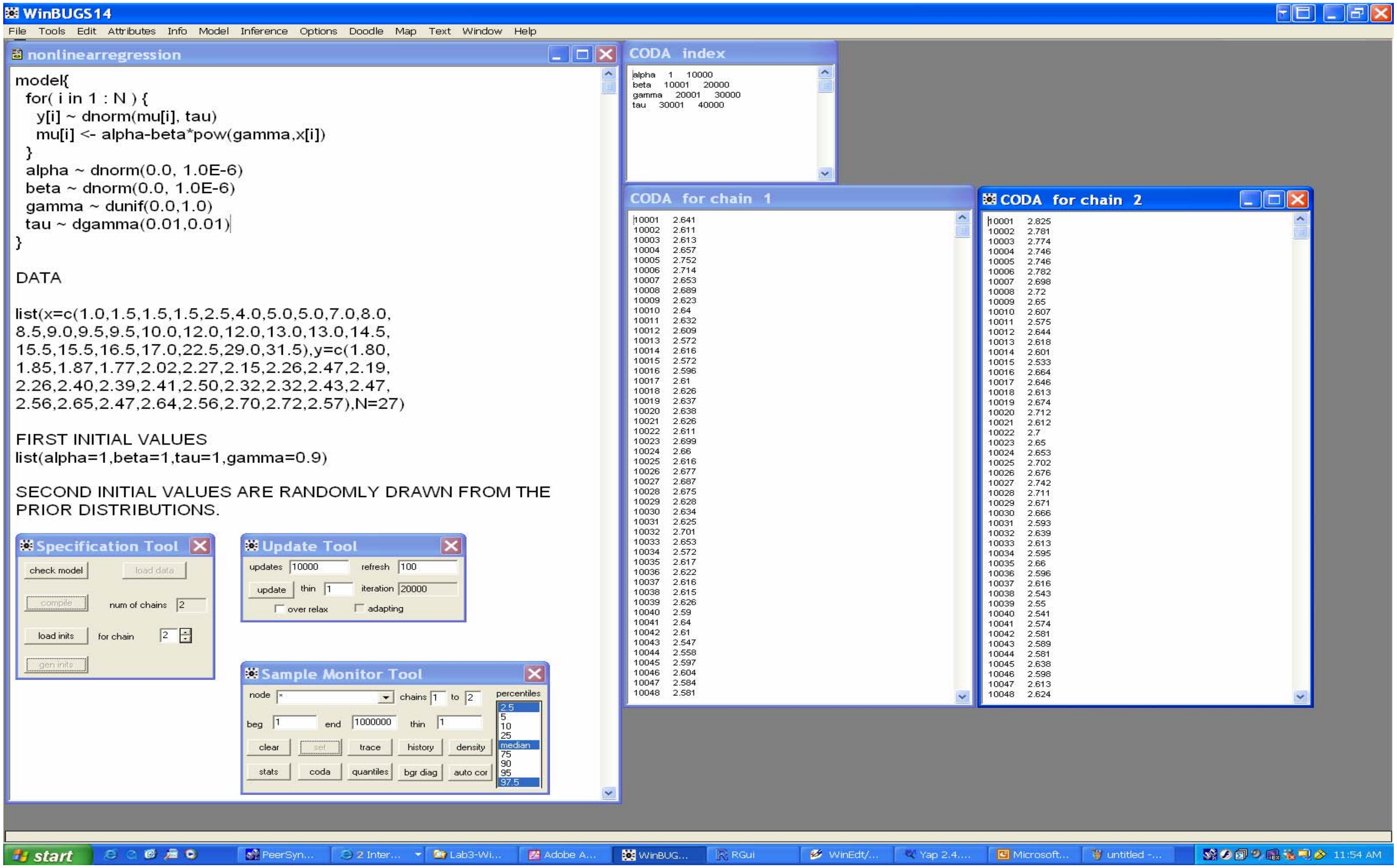
node * chains 1 to 2

beg 1 end 1000000 thin 1

percentiles 2.5 5 10 25 median 75 90 95 97.5

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

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.



nonlinearregression

```
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.

Specification Tool: num of chains 2; for chain 2

Update Tool: updates 10000; refresh 100; thin 1; iteration 20000; □ over relax; □ adapting

Sample Monitor Tool: node *; chains 1 to 2; beg 1; end 1000000; thin 1; percentiles: 2.5, 5, 10, 25, median, 75, 90, 95, 97.5

CODA index

```
alpha   1       10000
beta    10001   20000
gamma   20001   30000
tau     30001   40000
```

CODA for chain 1

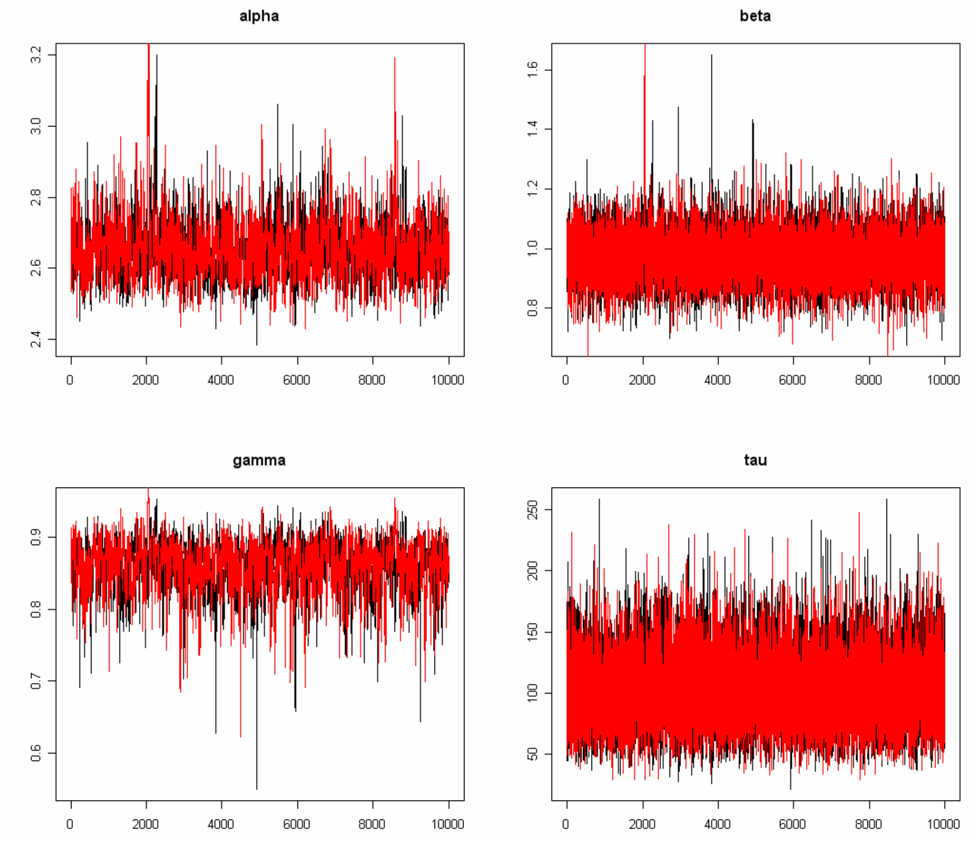
```
10001   2.641
10002   2.611
10003   2.613
10004   2.657
10005   2.752
10006   2.714
10007   2.653
10008   2.689
10009   2.623
10010   2.64
10011   2.632
10012   2.609
10013   2.572
10014   2.616
10015   2.572
10016   2.596
10017   2.61
10018   2.626
10019   2.637
10020   2.638
10021   2.626
10022   2.611
10023   2.699
10024   2.66
10025   2.616
10026   2.677
10027   2.687
10028   2.675
10029   2.628
10030   2.634
10031   2.625
10032   2.701
10033   2.653
10034   2.572
10035   2.617
10036   2.622
10037   2.616
10038   2.615
10039   2.626
10040   2.59
10041   2.64
10042   2.61
10043   2.547
10044   2.558
10045   2.597
10046   2.604
10047   2.584
10048   2.581
```

CODA for chain 2

```
10001   2.825
10002   2.781
10003   2.774
10004   2.746
10005   2.746
10006   2.782
10007   2.698
10008   2.72
10009   2.65
10010   2.607
10011   2.575
10012   2.644
10013   2.618
10014   2.601
10015   2.533
10016   2.664
10017   2.646
10018   2.613
10019   2.674
10020   2.712
10021   2.612
10022   2.7
10023   2.65
10024   2.653
10025   2.702
10026   2.676
10027   2.742
10028   2.711
10029   2.671
10030   2.666
10031   2.593
10032   2.639
10033   2.613
10034   2.595
10035   2.66
10036   2.596
10037   2.616
10038   2.543
10039   2.55
10040   2.541
10041   2.574
10042   2.581
10043   2.589
10044   2.581
10045   2.638
10046   2.598
10047   2.613
10048   2.624
```

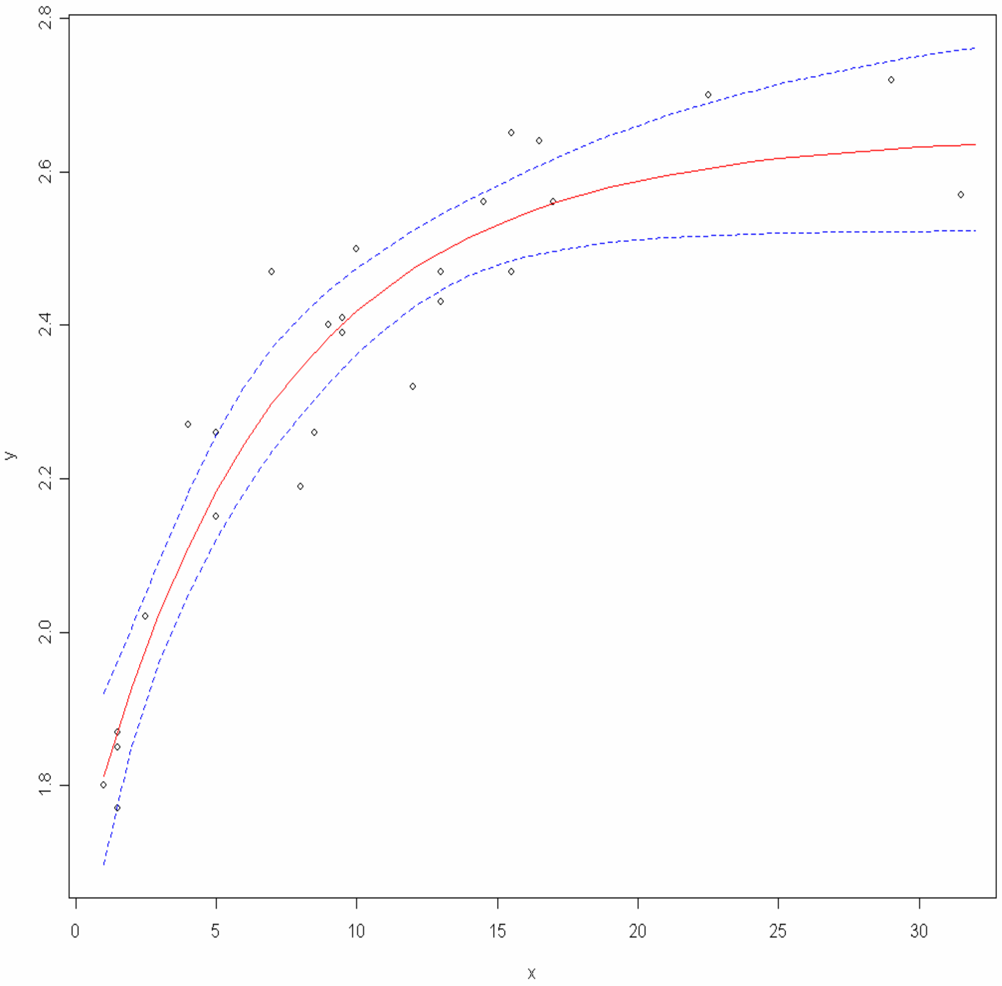
Suppose you save the two chains in files chain1.txt and chain2.txt

```
#########################################################
#  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 valus 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)
```



2.8
2.6
2.4
2.2
2.0
1.8
y
0
5
10
15
20
25
30
x