Basics and Issues with the BUGS Modeling Language

JEFF GILL
Distinguished Professor
Departments of Government and Mathematics & Statistics
American University

Underlying Technical Issues

- Technical notes:
  - The underlying engine is Adaptive Rejection Sampling (Gilks 1992, Gilks and Wild 1992), an MCMC implementation of rejection sampling.
  - There is a an option to switch to *Slice Sampling*.
  - This means that all priors and all likelihood functions must be either: (1) discrete, (2) conjugate, or (3) log concave.
  - Not a big deal as all GLMs with canonical link functions are well-behaved in this respect.
  - It also means that deterministic nodes must be linear functions.
  - Often these restrictions can be finessed.
  - `BUGS` likes simple, clean model structure. So if pre-processing in `R` can be done, it generally helps.

Running jags: Blood Pressure Example

- Creating a JAGS datafile to be called blood.pressure.jags.dat looks like this:

```
N <- 15
pressure <- c(132,155,130,142,150,128,126,118,180,124,150,134,140,142,128)
age.years <- c(49,56,52,46,57,42,43,45,56,52,42,57,56,56,53)
weight.lb <- c(145,216,115,170,172,166,164,152,275,221,175,132,188,178,168)
```

- The model in the file blood.pressure.jags is defined as:

```
model {
    for (i in 1:N) {
        mu[i] <- alpha + beta1*age.years[i] + beta2*weight.lb[i];
        pressure[i] ~ dnorm(mu[i],tau)
    }
    alpha  ~  dnorm(0.0,0.001)
    beta1  ~  dnorm(0.0,0.001)
    beta2  ~  dnorm(0.0,0.001)
    tau    ~  dgamma(2.0,0.1)
}
```

Running jags: Blood Pressure Example

- Put the intital values in the file `blood.pressure.jags.init`

```
list("alpha" = 0.0, "beta1" = 0.0, "beta2" = 0.0, "tau" = 1.0)
```

- The code for a JAGS window is:

```
model in "blood.pressure.jags"
data in "blood.pressure.jags.dat"
compile
inits in "blood.pressure.jags.init"
initialize
update 200000
monitor set alpha
monitor set beta1
monitor set beta2
monitor set tau
update 200000
coda *
exit
```

Running jags: Blood Pressure Example

- The JAGS window shows the progress:

```
. Reading data file blood.pressure.jags.dat
. Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 93
. Reading initial values file blood.pressure.jags.init
. . Updating 200000
-------------------------------------| 200000
************************************** 100%
```

Running jags: Blood Pressure Example

- Load the "convergence diagnostics" package if necessary:

```
library(coda)
```

- A single (sometimes finicky) command reads in the samples using two files:

```
bp <- read.coda("/Users/jgill/Bugs/CODAchain1.txt",
                "/Users/jgill/Bugs/CODAindex.txt")
```

- Use any convenient R command to summarize at this point:

```
apply(bp,2,summary)
        alpha  beta1  beta2     tau
Min.    -61.1 -2.980 -0.354 0.00109
1st Qu.  44.7  0.322  0.198 0.00724
Median   61.8  0.654  0.248 0.00933
Mean     61.9  0.654  0.247 0.00973
3rd Qu.  78.6  0.993  0.298 0.01180
Max.    218.0  3.150  0.722 0.03550
```

Blood Pressure Example from R, Setup

- Load required packages:

```
lapply(c("rjags","R2jags","arm","coda","superdiag","R2WinBUGS"),library,
    character.only=TRUE)
```

- Specify the data as a list in R:

```
bp.data <- list("N" = 15,
    "pressure" = c(132,155,130,142,150,128,126,118,180,124,150,134,140,
        142,128),
    "age.years" = c(49,56,52,46,57,42,43,45,56,52,42,57,56,56,53),
    "weight.lb" = c(145,216,115,170,172,166,164,152,275,221,175,132,
        188,178,168))
```

- Specify the Markov chain starting (initial) values:

```
bp.inits <- function(){ list("alpha" = 0.0, "beta1" = 0.0,
    "beta2"=0.0, "tau" = 1.0) }
```

- We need to say what parameters we are interested in:

```
bp.params <-  c("alpha","beta1", "beta2","tau")
```

Blood Pressure Example from R, Model

- The model is stipulated as a function:

```
bp.model <- function() {
    for (i in 1:N) {
        mu[i] <- alpha + beta1*age.years[i] + beta2*weight.lb[i];
        pressure[i] ~ dnorm(mu[i],tau)
    }
    alpha  ~  dnorm(0.0,0.001)
    beta1  ~  dnorm(0.0,0.001)
    beta2  ~  dnorm(0.0,0.001)
    tau    ~  dgamma(2.0,0.1)
}
```

- Run the model with the jags command:

```
bp.out <- jags(data=bp.data, inits=bp.inits, bp.params, n.iter=200000,
    model=bp.model)
module glm loaded
  |**************************************************| 100%
```

Simple but Real Example

- The Study: 1180 children in Florida who visited their HMO clinic and did or did not require an emergency room visit shortly thereafter.
- The data:
  - `erodd`, the dichotomous outcome variable, $[0, 1]$, indicating whether or not there was an emergency room visit.
  - `metq`, a severity score, $[1, 2, 3]$, indicating the degree of illness diagnosed at the HMO visit.
  - `np`, indication of profit, $[1]$, or nonprofit, $[-1]$, status of the HMO. This is the key explanatory variable of interest.

Simple but Real Example, Definition

- The statistical model:

$$\begin{aligned} erodd_i &\sim \mathcal{BE}(p_i) \\ \text{logit}(p_i) &= \alpha_0 + \alpha_1 metq_i + \alpha_2 np_i \\ \alpha_0 &\sim \mathcal{N}(0, 10) \\ \alpha_1 &\sim \mathcal{N}(0, 10) \\ \alpha_2 &\sim \mathcal{N}(0, 10) \end{aligned}$$

- Where the results of interest are the posterior distributions of α_0, α_1, and α_2.
- The model is run with a burn-in period of 1,000 cycles and the run for 11,000 more.

Simple but Real Example (cont.)

- The WinBUGS Code:

```
model
{
    for( i in 1 : N ) {
        logit(p[i]) <- alpha0 + alpha1 * metq[i] + alpha2 * np[i]
        erodd[i] ~ dbern(p[i])
    }
    alpha0 ~ dnorm(0.0,0.1)
    alpha1 ~ dnorm(0.0,0.1)
    alpha2 ~ dnorm(0.0,0.1)
}

list(alpha0 = 0, alpha1 = 0, alpha2 = 0)

list(N=1180, erodd = c(1,1,1,0,1,0,1,...), metq = c(3,3,3,3,3,2,3,...),
     np = c(-1,-1,-1,1,-1,-1,1,...))
```

Simple but Real Example (cont.)

- The Results:

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
alpha0	-1.971	0.22280	0.008254	-2.41300	-1.968	-1.540	1	11000
alpha1	0.2808	0.09423	0.003505	0.098070	0.2803	0.4645	1	11000
alpha2	0.1646	0.08042	9.194E-4	0.008893	0.1639	0.3213	1	11000

Thermonuclear Testing Example

- There were certainly many political, economic, engineering, and policy reasons for conducting nuclear tests in the United States, we will specify a model with only three predictors for each year: U.S. gross domestic product (in trillions of dollars, base 1996), U.S domestic non-financial debt (in trillions of dollars, base 1996, outstanding credit market debt of federal, state, and local government as well as private non-financial sectors), and counts of Soviet tests.
- Specify a simple Bayesian generalized linear model with a Poisson link function according to:

```
model {
    for (i in 1:n)  {
        log(mu[i]) <- b0 + b1*US.GDP[i] + b2*US.DEBT[i]
                         + b3*(SOV.TEST[i]-mean(SOV.TEST))
        US.TEST[i] ~ dpois(mu[i])
    }

    b0 ~ dnorm(0,1.0E-4)
    b1 ~ dnorm(0,1.0E-4)
    b2 ~ dnorm(0,1.0E-4)
    b3 ~ dunif(-100,100)
}
```

Thermonuclear Testing Example

- JAGS commands (I usually cut-and-paste from a file):

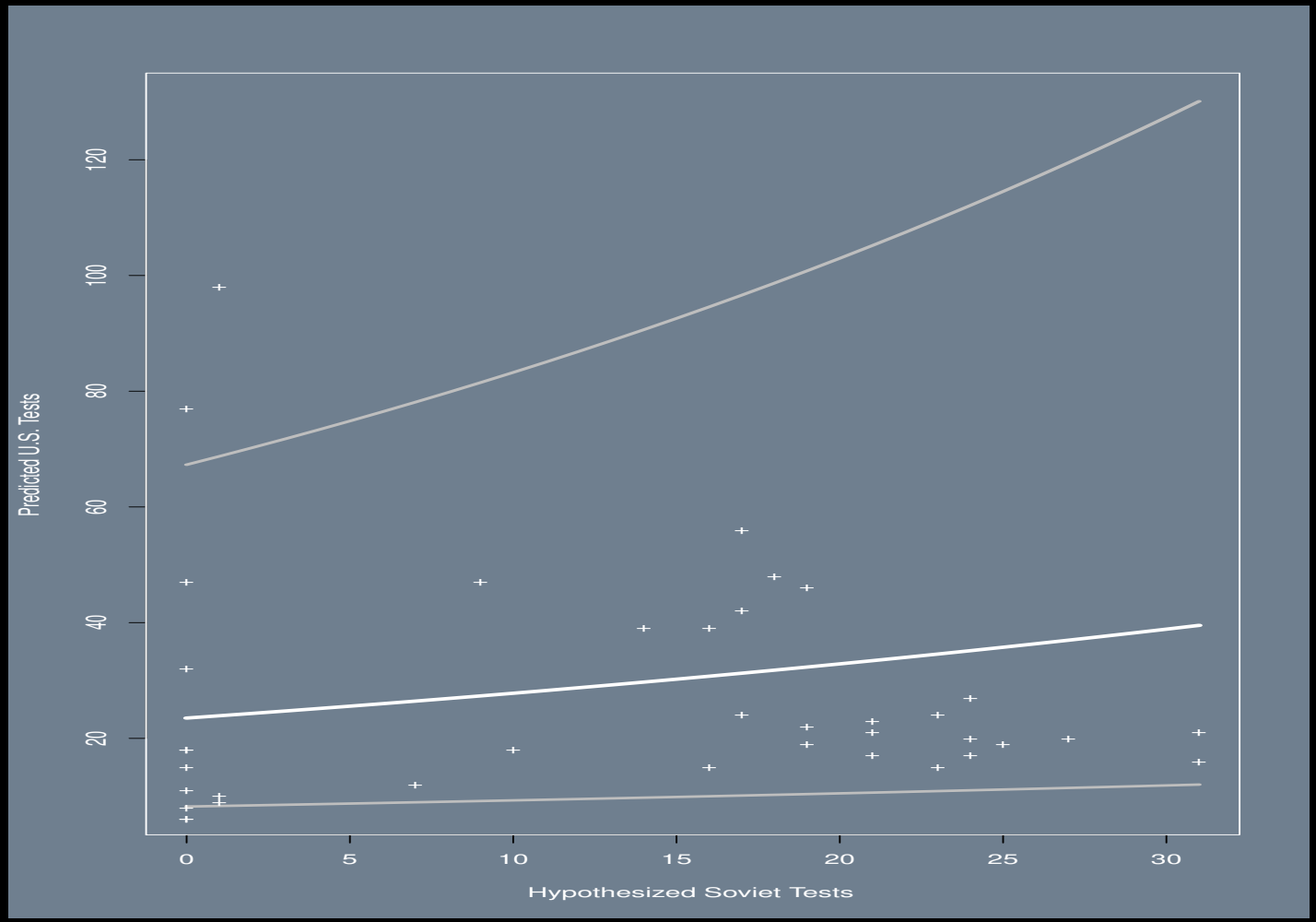
```
model in "nukes2.bug"
data in "nukes2.data"
compile
inits in "nukes2.init"
initialize
update 500000
monitor set b0
monitor set b1
monitor set b2
monitor set b3
update 1000000
coda *
exit
```

- Looking at the JAGS commands that go into the JAGS window is illustrative since this is still the underlying process when we call JAGS for R.

Thermonuclear Testing Example, Summary of Results

			Posterior Quantiles				
	Mean	SD	2.5%	25%	50%	75%	97.5%
CONSTANT	3.835	0.078	3.683	3.783	3.836	3.888	3.988
US.GDP	-0.903	0.250	-1.398	-1.073	-0.902	-0.734	-0.412
US.DEBT	0.345	0.131	0.088	0.257	0.345	0.434	0.603
SOV.TEST	0.017	0.005	0.008	0.014	0.017	0.020	0.026

Posterior Prediction With 1 SE Bars



A Hierarchical Model of Lobbying Influence in the US States

- The American State Administrator's Project (ASAP) survey asks administrators about the influence of a variety of external political actors including "clientele groups" in their agencies.
- Clientele group is arguably not perfectly synonymous with interest group, but previous studies have used these terms interchangably (Kelleher and Yackee 2009).
- Consider a 713×22 matrix $\mathbf{X}$ with a leading column of 1's for individual level explanatory variables, and a 50×3 matrix $\mathbf{Z}$ for state-level explanatory variables.
- Our outcome variable (`group influence`) measures the respondents' perception of interest groups' influence on total budget, special budgets, and general public policies.
- We relate these variables through the linear hierarchical model:

$$Y_i \sim N(\boldsymbol{\alpha}_{j[i]} + \boldsymbol{\beta}\mathbf{X}_i, \sigma_y^2), \qquad \text{for } i = 1, \ldots, 713$$
$$\boldsymbol{\alpha}_j \sim N(\boldsymbol{\gamma}\mathbf{Z}, \sigma_\alpha^2), \qquad \text{for } j = 1, \ldots, 50,$$

  using hierarchical notation from Gelman and Hill (2007) to indicate that the ith respondent is nested in the jth state: $\boldsymbol{\alpha}_{j[i]}$ to get a state-specific random intercept.
- This random intercept is the parameterized at the second level by the three explanatory variables in $\mathbf{Z}$ and their corresponding estimated coefficients, $\boldsymbol{\gamma}$.

A Hierarchical Model of Lobbying Influence in the US States

- Note the two different variances accounted for in this specification. The term σ_y^2 measures the *within-state* variance of the outcome variables, whereas the term σ_α^2 gives the variance of the mean estimates *between-states*.
- Prior distributions are diffuse normals centered at the point estimates from Kelleher and Yackee (2009), Model 3 (page 593), for overlapping covariate use, but diffuse for others.
- We deviate from their values in only two ways: we substitute the variable `elected/board` for their variable Merit Position, and since we are using 2008 data only there cannot be a dummy variable for the year 2004.
- Thus:

$$\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_{ky}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}})$$
$$\boldsymbol{\gamma} \sim N(\boldsymbol{\gamma}_{ky}, \boldsymbol{\Sigma}_{\boldsymbol{\gamma}}),$$

  where the $\boldsymbol{\Sigma}_{\boldsymbol{\beta}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\gamma}}$ matrices are diagonal forms with large variances relative to the Kelleher and Yackee point estimates.

- Get the code from the syllabus.

A Hierarchical Model of Lobbying Influence in the US States, JAGS Code

```
model  {
    for (i in 1:SUBJECTS) {
        mu[i] <- alpha[state.id[i]]
            + beta[1]*contractind[i] + beta[2]*govinf3[i] + beta[3]*leginf3[i]
            + beta[4]*clientappt[i] + beta[5]*years[i] + beta[6]*gender[i]
            + beta[7]*educ[i] + beta[8]*party5pt[i] + beta[9]*X_Ifuncat13_2[i]
            + beta[10]*X_Ifuncat13_3[i] + beta[11]*X_Ifuncat13_4[i]
            + beta[12]*X_Ifuncat13_5[i] + beta[13]*X_Ifuncat13_6[i]
            + beta[14]*X_Ifuncat13_7[i] + beta[15]*X_Ifuncat13_8[i]
            + beta[16]*X_Ifuncat13_9[i] + beta[17]*X_Ifuncat13_10[i]
            + beta[18]*X_Ifuncat13_11[i] + beta[19]*X_Ifuncat13_12[i]
            + beta[20]*timegroupsmed[i] + beta[21]*timemedXcont[i]
        groupinf3[i] ~ dnorm(mu[i],tau)
    }
    for (j in 1:STATES) {
        eta[j] <- gamma[1]*gov6099all[j] + gamma[2]*totreg[j]
                + gamma[3]*nonprofs.per10k[j]
        alpha[j] ~ dnorm(eta[j],tau.alpha)
    }
```

A Hierarchical Model of Lobbying Influence in the US States, JAGS Code

```
    beta[1]   ~ dnorm(0.070,1)     # PRIOR MEANS FROM KELLEHER AND YACKEE 2009, MODEL 3
    beta[2]   ~ dnorm(-0.054,1)
    beta[3]   ~ dnorm(0.139,1)
    beta[4]   ~ dnorm(0.468,1)
    beta[5]   ~ dnorm(0.017,1)
    beta[6]   ~ dnorm(0.207,1)
    beta[7]   ~ dnorm(0.056,1)
    beta[8]   ~ dnorm(0.039,1)
    beta[9]   ~ dnorm(0.0,1)
    beta[10]  ~ dnorm(0.0,1)
    beta[11]  ~ dnorm(0.0,1)
    beta[12]  ~ dnorm(0.0,1)
    beta[13]  ~ dnorm(0.0,1)
    beta[14]  ~ dnorm(0.0,1)
    beta[15]  ~ dnorm(0.0,1)
    beta[16]  ~ dnorm(0.0,1)
    beta[17]  ~ dnorm(0.0,1)
    beta[18]  ~ dnorm(0.0,1)
    beta[19]  ~ dnorm(0.0,1)
    beta[20]  ~ dnorm(0.184,1)
    beta[21]  ~ dnorm(0.146,1)
    gamma[1]  ~ dnorm(0.0,1)
    gamma[2]  ~ dnorm(0.0,1)
    gamma[3]  ~ dnorm(0.0,1)
    tau       ~ dgamma(1.0,1)
    tau.alpha ~ dgamma(1.0,1)
}
```

A Hierarchical Model of Lobbying Influence in the US States

- Using `beta[ ]` is more efficient for running the model and importing the samples.
- If you have the same priors for these coefficients, then the priors can be specified in a loop:

```
for (k in 1:21) beta[k] ~ dnorm(0,1)
```

- JAGS specfies starting values in R vector and scalar forms:

```
tau <- 10
tau.alpha <- 10
alpha <- 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,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)
zeta <- 1
beta <- c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)
gamma <- c(1,1,1)
```

A Hierarchical Model of Lobbying Influence in the US States, Data File

```
asap.jags.list <- list(
state.id <-
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, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6,
:
2.9545, 9.9194, 0.8706, 0.5967, 3.9635, 3.5073, 1.1225, 3.4171,
0.4537),
STATES <- 50, SUBJECTS <- 713)
```

Using rjags from R, Data Handling (same model)

```
lapply(c("rjags","R2jags","arm","coda","superdiag","R2WinBUGS"),library,
       character.only=TRUE)

source("Article.JPART/asap.rjags.dat")  # READS IN THE LIST VERSION OF THE DATA

names(asap.jags.list) <- c("state.id","contracting","gov.influence","leg.influence",
  "elect.board","years.tenure","education","party.ID","category2","category3",
  "category4","category5","category6","category7", "category8","category9",
  "category10","category11","category12","med.time","medt.contr","grp.influence",
  "gov.ideology","lobbyists","nonprofits","STATES","SUBJECTS")
```

A Hierarchical Model of Lobbying Influence in the US States, Data in R

```
asap.jags.list
$state.id
 [1]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
[21]  1  1  1  1  1  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2
 :
[37]  2.1932  6.6298  0.7195  2.1484  0.6801  2.9545  9.9194  0.8706  0.5967
[46]  3.9635  3.5073  1.1225  3.4171  0.4537

$STATES
[1] 50

$SUBJECTS
[1] 713
```

Using rjags from R (same model)

```
# DEFINE THE MODEL
asap.model2.rjags  <- function()  {
for (i in 1:SUBJECTS) {
    mu[i] <- alpha[state.id[i]]
      + beta[1]*contracting[i] + beta[2]*gov.influence[i] + beta[3]*leg.influence[i]
      + beta[4]*elect.board[i] + beta[5]*years.tenure[i]  + beta[6]*education[i]
      + beta[7]*party.ID[i]    + beta[8]*category2[i]     + beta[9]*category3[i]
      + beta[10]*category4[i]  + beta[11]*category5[i]    + beta[12]*category6[i]
      + beta[13]*category7[i]  + beta[14]*category8[i]    + beta[15]*category9[i]
      + beta[16]*category10[i] + beta[17]*category11[i]   + beta[18]*category12[i]
      + beta[19]*med.time[i]   + beta[20]*medt.contr[i]
      grp.influence[i] ~ dnorm(mu[i],tau.y)
    }
    for (j in 1:STATES) {
      eta[j]  <- gamma[1]*gov.ideology[j]+gamma[2]*lobbyists[j]+gamma[3]*nonprofits[j]
      alpha[j] ~ dnorm(eta[j],tau.alpha)
    }
    beta[1]   ~ dnorm(0.070,1)  # PRIOR MEANS FROM KELLEHER AND YACKEE 2009, MODEL 3
    beta[2]   ~ dnorm(-0.054,1)
```

```
    beta[3]   ~ dnorm(0.139,1)
    beta[4]   ~ dnorm(0.051,1)
    beta[5]   ~ dnorm(0.017,1)
    beta[6]   ~ dnorm(0.056,1)
    beta[7]   ~ dnorm(0.039,1)
    beta[8]   ~ dnorm(0.0,1)    # DIFFUSE PRIORS
    beta[9]   ~ dnorm(0.0,1)
    beta[10]  ~ dnorm(0.0,1)
    beta[11]  ~ dnorm(0.0,1)
    beta[12]  ~ dnorm(0.0,1)
    beta[13]  ~ dnorm(0.0,1)
    beta[14]  ~ dnorm(0.0,1)
    beta[15]  ~ dnorm(0.0,1)
    beta[16]  ~ dnorm(0.0,1)
    beta[17]  ~ dnorm(0.0,1)
    beta[18]  ~ dnorm(0.0,1)
    beta[19]  ~ dnorm(0.184,1)  # PRIOR MEANS FROM KELLEHER AND YACKEE 2009, MODEL 3
    beta[20]  ~ dnorm(0.156,1)
    gamma[1]  ~ dnorm(0.0,1)    # DIFFUSE PRIORS
    gamma[2]  ~ dnorm(0.0,1)
    gamma[3]  ~ dnorm(0.0,1)
```

```
    tau.y     ~ dgamma(1.0,1)
    tau.alpha ~ dgamma(1.0,1)
}
```

Using rjags from R (same model)

```
# SETUP INITIAL VALUES AND PARAMETER NAMES
asap.inits <- function(){ list("tau.y" = 10, "tau.alpha" = 10,
    "beta" = rep(1,20), "gamma" = c(1,1,1)) }
asap.params <- c("beta","gamma","tau.y","tau.alpha")

# RUN THE SAMPLER AND COLLECT coda SAMPLES
asap.out <- jags(data=asap.jags.list, inits=asap.inits, asap.params,
    n.iter=5000, model="Article.JPART/asap.model2.rjags", DIC=TRUE)

# USE THE autojags() FUNCTION TO RUN UNTIL G&R < 1.1
#asap.out2 <- autojags(asap.out)

# OR update() TO RUN LONGER
asap.out3 <- update(asap.out, n.iter=200000)

# GET SUMMARY OF SAMPLES
summary(asap3.mcmc)
```

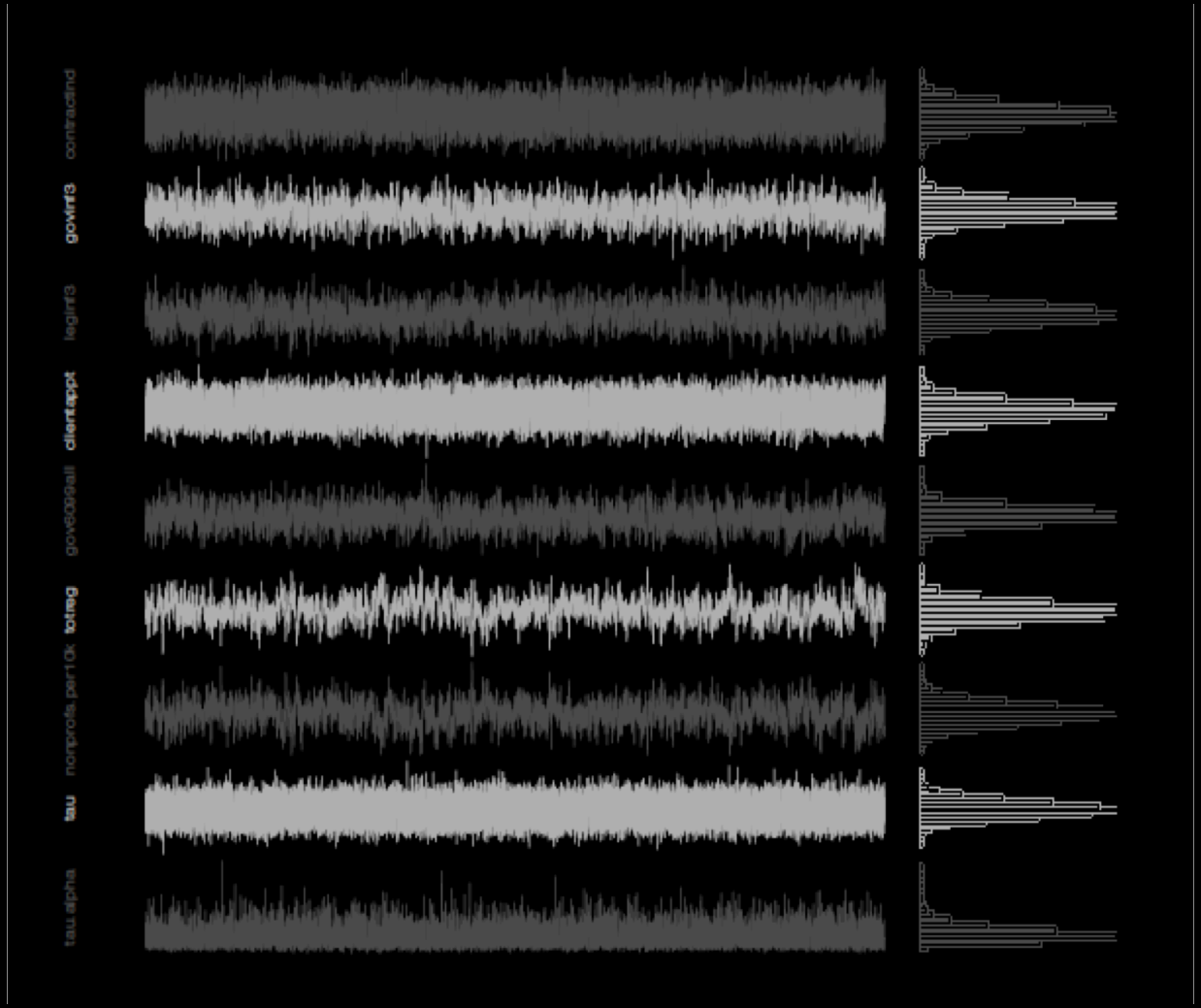
A Hierarchical Model of Lobbying Influence in the US States, Results

Parameters	Post.Mean	Post.SE	95%.Lower.CI	95%.Upper.CI
α mean(1:50)	1.4502	0.7076	0.0632	2.8372
contractind	0.1411	0.0966	-0.0482	0.3304
govinf3	0.0416	0.0379	-0.0327	0.1159
leginf3	0.3507	0.0403	0.2717	0.4296
clientappt	1.2318	0.3515	0.5428	1.9207
years	0.0508	0.0240	0.0038	0.0978
gender	0.5334	0.3038	-0.0620	1.1288
educ	0.0630	0.1214	-0.1749	0.3010
party5pt	0.0373	0.0844	-0.1283	0.2028
X_Ifuncat13_2	-0.3631	0.5386	-1.4187	0.6926
X_Ifuncat13_3	-0.1172	0.5860	-1.2658	1.0314
X_Ifuncat13_4	1.5814	0.4612	0.6774	2.4854
X_Ifuncat13_5	-0.3687	0.5001	-1.3489	0.6116
X_Ifuncat13_6	1.1478	0.5385	0.0923	2.2033
X_Ifuncat13_7	1.7630	0.4336	0.9131	2.6128
X_Ifuncat13_8	0.9254	0.4965	-0.0478	1.8986
X_Ifuncat13_9	0.9789	0.4844	0.0295	1.9284

A Hierarchical Model of Lobbying Influence in the US States, Results

Parameters	Post.Mean	Post.SE	95%.Lower.CI	95%.Upper.CI
X_Ifuncat13_10	0.4335	0.4608	-0.4698	1.3367
X_Ifuncat13_11	-0.0547	0.4239	-0.8856	0.7761
X_Ifuncat13_12	-0.5667	0.5703	-1.6846	0.5512
timegroupsmed	1.0671	0.3536	0.3741	1.7602
timemedXcont	-0.0602	0.1353	-0.3254	0.2050
gov6099all	0.0189	0.0063	0.0066	0.0312
totreg	0.0006	0.0007	-0.0008	0.0021
nonprofs.per10k	0.0000	0.0000	-0.0000	0.0000
τ	0.0774	0.0042	0.0691	0.0857
τ_α	3.0344	1.3206	0.4460	5.6228

A Hierarchical Model of Lobbying Influence in the US States, Convergence Plots



JAGS Versus BUGS

- Even if you use JAGS you need the R2WinBUGS package if you want to use write.model.
- JAGS uses the R dump() format for data and initial values, whereas WinBUGS uses the R list format.
- Use writeDatafileR.R to create BUGS data from R and bugs2jags to convert data to JAGS .
- The biggest difference is handling censoring and truncation.
- In WinBUGS the I(,) construct is used in WinBUGS for:
  - ▷ censoring as a posterior restriction,
  - ▷ truncation of top-level parameters as a prior restriction,
  - ▷ but it is invalid to use it for truncation of a distribution with unobserved parameters.
- Format:

```
z ~ ddist(beta)I(lower, upper).
```

JAGS Versus BUGS , Truncation

- JAGS uses the T(,) construct for truncation.
- Normally it looks something like this:

  ```
  beta[2] ~ dnorm(0,0.1)T[L,U]
  ```

  where L and U are lower and upper bounds that you select. You can also only use one: T(L,) or T(,U).
- Note that in calling JAGS from R you need make the adjustment:

  ```
  beta[2] ~ dnorm(0,0.1);T[L,U]
  ```

JAGS Versus BUGS

- Censoring in JAGS is represented by the novel distribution `dinterval`.
- First define an indicator vector for the variable, say `is.censored`, where for each value `is.censored[i]=0` if it is noncensored and `is.censored[i]=1` if it is censored.
- The variable itself, say `X1`, contains data values for the noncensored and `NA` for the censored.
- These are related in the model by: `is.censored ~ dinterval(X1,censor.limit)` for a single censoring point, or `is.censored ~ dinterval(X1,censor.limit[i])` for variable censoring points.
- Then JAGS imputes a random value for `X1` from the likelihood function you specify.
- From the JAGS manual, an example of right-hand failure time censoring in a loop:

```
is.censored[i] ~ dinterval(t[i], t.censor[i])
t[i] ~ dweib(r,mu[i])
```

where: `t.censor[i]` is a censoring time, `is.censored[i]` is a censoring indicator for the ith case, `t[i]` is a failure time.

Left Censoring Example In JAGS

```
bounds.data                  L        U
               [1,] 14.98266 15.68029
               [2,] NA       21.91590
               [3,] 18.34953 19.04716....

censor.model <- function() {
    for (i in 1:50) {
        is.censored[i] ~ dinterval(t[i], bounds.data[i,])
        t[i] ~ dnorm(mu,tau)
    }
    mu ~ dnorm(0,0.01)
    tau ~ dgamma(1,0.01)
}

write.model(censor.model,local.model); data <- list("bounds.data"=bounds.data)
inits <- list(mu=rnorm(1),tau=rgamma(1,1,.01),t=as.vector(apply(bounds.data,1,mean)))
censor.out <- jags(data,inits, c("mu","tau"), model.file="local.model", n.chains=5,
                   n.iter=50000,n.burnin=10000, n.thin=1, DIC=TRUE,
                   working.directory=NULL,refresh = 50000/50, progress.bar = "text")
```

Inference and the DIC

- The Deviance Information Criterion is an alternative to AIC that explicitly accounts for the different role that parameters can play in a multilevel model (parent nodes, descendent nodes, etc.).
- See Spielgelhalter, Best, Carlin, and van der Linde, "Bayesian Measures of Model Complexity and Fit." JRSS-B 64 (2002), 583-639).
- A rough approximation to theoretical out-of-sample predictive error.
- Note: make sure the chains have converged first.
- Two objectives: describing model complexity, giving model fit comparisons.
- Calculated for you by `WinBUGS` and `JAGS` , so you do not have to do the hard work.

The Deviance Information Criterion (DIC)

- Model under consideration, defined by the likelihood $p(\mathbf{y}|\boldsymbol{\theta})$ for data $\mathbf{y}$ and parameter vector $\boldsymbol{\theta}$.
- A measure of "Bayesian deviance" is given by:

$$D(\boldsymbol{\theta}) = -2\log[p(\mathbf{y}|\boldsymbol{\theta})] + 2\log[f(\mathbf{y})]$$

  where $f(\mathbf{y})$ is some function of just the data, the "standardizing factor."
- Note the similarity to the Akaike Information Criterion:

$$\text{AIC} = -2\ell(\hat{\boldsymbol{\theta}}|\boldsymbol{y}) + 2p$$

  and the Bayesian Information Criterion:

$$\text{BIC} = -2\ell(\hat{\boldsymbol{\theta}}|\boldsymbol{y}) + \log(n)p.$$

- Spiegelhalter et al. de-emphasize $f(\mathbf{y})$ for model comparison and even suggest using $f(\mathbf{y}) = 1$ (giving zero contribution above) since this term must be identical for both model calculations and therefore cancels out.
- So if you had a point estimate of $\boldsymbol{\theta}$, then $D(\boldsymbol{\theta})$ would be the deviance from the saturated model.

The Deviance Information Criterion (DIC)

- We can use the deviance in explicitly posterior terms by inserting a condition on the data and taking an expectation over $\boldsymbol{\theta}$, to get the *Expected Bayesian Deviance*:

$$\begin{aligned}\overline{D(\boldsymbol{\theta})} &= E_{\boldsymbol{\theta}}[-2\log[p(\mathbf{y}|\boldsymbol{\theta})|\mathbf{y}] + 2\log[f(\mathbf{y})] \\ &= -2\int_{\Theta}(\log[p(\mathbf{y}|\boldsymbol{\theta})] - \log[f(\mathbf{y})])\,d\theta,\end{aligned}$$

- Now we do *not* need a single point estimate of $\boldsymbol{\theta}$.
- This is a posterior mean difference that now gives a measure of Bayesian model fit.
- The conditioning on the data is easy, but the expectation might be hard analytically.
- This is usually easy in an MCMC context since we get the expectation empirically from the samples.

The Deviance Information Criterion (DIC), p_D

- Define $\tilde{\boldsymbol{\theta}}$ as any posterior estimate of $\boldsymbol{\theta}$, which can be the posterior mean or some other easily produced value.
- We can also insert $\tilde{\boldsymbol{\theta}}$ into $D(\boldsymbol{\theta})$ to get the "plug-in" deviance.
- The *effective dimension* of the model is now defined by:

$$p_D = \overline{D(\boldsymbol{\theta})} - D(\tilde{\boldsymbol{\theta}}),$$

  where this is the "mean deviance minus the deviance of the means."
- So p_D is the weighted sum of the parameters in which each is weighted according to:
  - $\omega_p = 1$ for parameters unconstrained by prior information
  - $\omega = 0$ for parameters completely specified (fixed) by prior information
  - $\omega \in [0{:}1]$ for parameters with specific dependencies on the data or priors.
- p_D is the most theoretically important principle here since it improves the AIC method of weighting all p model parameters equally in a non-hierarchical model specification.

The Deviance Information Criterion (DIC), Linear Models

- For regular normal linear models $p_D = \text{tr}(H)$, where H is the hat matrix ($Hy = \hat{y}$, projecting data onto fitted values).
- Since $p_D = \sum_{i=1}^{n} h_{ii}$, it is also the sum of the data point leverages in this linear context.
- This is more complicated in nonlinear hierarchical linear models, but also more useful.
- Generally the hat matrix, $\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$, is not relevant for interesting multilevel model specifications.

The Deviance Information Criterion (DIC)

- Due to the subtraction, p_D is independent of our choice of $f(\mathbf{y})$:

$$
\begin{aligned}
p_D &= \overline{D(\boldsymbol{\theta})} - D(\tilde{\boldsymbol{\theta}}) \\
&= \left\{ E_{\boldsymbol{\theta}}[-2\log[p(\mathbf{y}|\boldsymbol{\theta})|\mathbf{y}] + 2\log[f(\mathbf{y})] \right\} - \left\{ -2\log[p(\mathbf{y}|\tilde{\boldsymbol{\theta}})] + 2\log[f(\mathbf{y})] \right\} \\
&= E_{\boldsymbol{\theta}|\mathbf{y}}[-2\log(p(\mathbf{y}|\boldsymbol{\theta}))] + 2\log(p(\mathbf{y}|\tilde{\boldsymbol{\theta}})) \\
&= E_{\boldsymbol{\theta}|\mathbf{y}}\left[-2\log\left(\frac{p(\boldsymbol{\theta}|\mathbf{y})p(\mathbf{y})}{p(\boldsymbol{\theta})} \right) \right] + 2\log\left(\frac{p(\tilde{\boldsymbol{\theta}}|\mathbf{y})p(\mathbf{y})}{p(\tilde{\boldsymbol{\theta}})} \right) \qquad \text{(Bayes' Law)} \\
&= 2E_{\boldsymbol{\theta}|\mathbf{y}}\left[-\log\left(\frac{p(\boldsymbol{\theta}|\mathbf{y})}{p(\boldsymbol{\theta})} \right) \right] + \log\left(\frac{p(\tilde{\boldsymbol{\theta}}|\mathbf{y})}{p(\tilde{\boldsymbol{\theta}})} \right).
\end{aligned}
$$

- The ratio $p(\boldsymbol{\theta}|\mathbf{y})/p(\boldsymbol{\theta})$ here is the gain in information provided by conditioning on the data in the model, and correspondingly $p(\tilde{\boldsymbol{\theta}}|\mathbf{y})/p(\tilde{\boldsymbol{\theta}})$ is the gain in information plugging-in the chosen estimate.

The Deviance Information Criterion (DIC)

- In a simulation context, p_D can be computed as the mean of simulated values of $D(\tilde{\boldsymbol{\theta}})$ minus $D(\boldsymbol{\theta})$ plugging the mean of the simulated values of $\boldsymbol{\theta}$.
- So while $\overline{D(\boldsymbol{\theta})}$ is a Bayesian measure of model fit, p_D is designed to be a "complexity measure" for the effective number of parameters in the model.
- The Deviance Information Criterion is created by differencing these terms:

$$\begin{aligned} DIC &= \overline{D(\boldsymbol{\theta})} + p_D \\ &= \overline{D(\boldsymbol{\theta})} + \left(\overline{D(\boldsymbol{\theta})} - D(\tilde{\boldsymbol{\theta}})\right) \\ &= 2\overline{D(\boldsymbol{\theta})} - D(\tilde{\boldsymbol{\theta}}) \end{aligned}$$

  and is thus the Bayesian measure of model fit above with an extra complexity penalization.
- Models with *smaller* DIC are said to *fit better*.

Worries About the DIC

- When the posterior distributions in non-symmetric or multimodal, the use of the posterior mean may be inappropriate.
- p_D is not invariant to reparameterization of $\boldsymbol{\theta}$.
- p_D can be negative, so DIC can also be negative.
- DIC is not a function of the marginal likelihood, therefore it is not related to Bayes Factor comparisons.
- DIC does not work when the likelihood depends on discrete parameters.
- So it does not work with mixture likelihoods (`WinBUGS` will "grey-out" the DIC button automatically).
- Models are penalized for "intermediate" variable definitions in `BUGS` .

The Deviance Information Criterion, Example

- Western and Garip (2005) use the example of village-level migration to contrast hierarchical model specifications for a dichotomous outcome.
- Survey data on young adults (18-25) in 22 Northeastern Thai villages where the outcome variable is 1 if the respondent spent at least two months away from the village in 1990 and 0 otherwise.
- Individual-level explanatory variables are:
  - sex
  - age
  - years of education
  - number of prior trips
- The village-level explanatory variables are:
  - priors trips from the villages
  - the Gini index of prior trips.
- There are two data matrices here with rows: x_{ij} for individual i in village j, and z_j for village j.

The Deviance Information Criterion, Example

Their Name	Specification	Prior Parameters
Pooled	$\text{logit}(p_{ij}) = \alpha + x'_{ij}\beta + z'_j\gamma$	$\alpha \sim \mathcal{N}(0, 10^6)$ $\beta[1:4] \sim \mathcal{N}(0, 10^6)$ $\gamma[1:2] \sim \mathcal{N}(0, 10^6)$
Fixed Effect	$\text{logit}(p_{ij}) = \alpha_j + x'_{ij}\beta$	$\alpha[1:22] \sim \mathcal{N}(0, 10^6)$ $\beta[1:4] \sim \mathcal{N}(0, 10^6)$
Random Effect	$\text{logit}(p_{ij}) = \alpha_j + x'_{ij}\beta + z'_j\gamma$ $\alpha[1:22] \sim \mathcal{N}(\mu, \tau^2)$	$\mu = 0$ $\tau \sim \mathcal{IG}(10^{-3}, 10^{-3})$ $\beta[1:4] \sim \mathcal{N}(0, 10^6)$ $\gamma[1:2] \sim \mathcal{N}(0, 10^6)$
Random Intercept and Random Slope	$\text{logit}(p_{ij}) = \alpha_j + x'_{ij}\beta_j + z'_j\gamma$ $\alpha[1:22] \sim \mathcal{N}(\mu_\alpha, \tau^2_\alpha)$ $\beta[1:22][1:4] \sim \mathcal{N}(\mu_\beta, \tau^2_\beta)$	$\mu_\alpha \sim \mathcal{N}(0, 10^6)$ $\tau^2_\alpha \sim \mathcal{IG}(10^{-3}, 10^{-3})$ $\mu_\beta \sim \mathcal{N}(0, 10^6)$ $\tau^2_\beta \sim \mathcal{IG}(10^{-3}, 10^{-3})$ $\gamma[1:2] \sim \mathcal{N}(0, 10^6)$

The Deviance Information Criterion, Example

Table 1: Model Comparison, Thai Migration

Model:	*Pooled*	*Fixed Effect*	*Random Effect*	*Rdm. Slope & Intercept*
Intercept	-0.80 (0.12)	-0.66 (0.10)	-0.79 (0.13)	-0.76 (0.14)
Male	0.32 (0.14)	0.38 (0.14)	0.33 (0.14)	0.30 (0.15)
Age	-0.13 (0.08)	-0.15 (0.08)	-0.13 (0.08)	-0.13 (0.09)
Education	0.39 (0.07)	0.38 (0.08)	0.39 (0.07)	0.39 (0.08)
Prior trips(i)	1.08 (0.09)	1.13 (0.10)	1.08 (0.09)	1.22 (0.16)
Prior trips(j)	-0.61 (0.37)		-0.59 (0.43)	-.69 (0.46)
Gini trips(j)	-0.62 (0.21)		-0.60 (0.24)	-0.61 (0.25)
Posterior means (posterior standard deviations in parentheses)				
p_D	7.02	26.24	10.93	30.58
DIC	1259.41	1273.29	1259.62	1247.55

- The individual- and village-level coefficients are remarkably similar, except of course that the fixed effects model does not have village-level coefficients).
- Posterior distributions for the random intercept and random slope/random intercept are slightly more diffuse than for the pooled model, reflecting the random effects modeling heterogeneity.

The Deviance Information Criterion, Example Observations

Model:	*Pooled*	*Fixed Effect*	*Random Effect*	*Rdm. Slope & Intercept*
p_D	7.02	26.24	10.93	30.58
DIC	1259.41	1273.29	1259.62	1247.55

- The $p_D = 7.02$ value shows that the *Pooled Model* is the most parsimonious, with the extra 0.02 above the $1 + 6$ specified parameters coming from prior information.
- The *Fixed Effect Model* has $4 + 22$ parameters, but a high $p_D = 26.24$.
- Even though it contains $1 + 6 + 22$ parameters, the *Random Effect Model* is only slightly less parsimonious than the *Pooled Model* with $p_D = 10.93$.
- The *Random Intercept/Random Slope Model* is the least parsimonious with $22 + 4 \times 22 = 110$ random effects and 2 village-level parameters specified, which is only moderately less parsimonious than the random effect model with 26 parameters, justifying the hierarchical structure imposed.

Some More Qualifications On The DIC

- A lack of invariance to reparameterizations of the parameters where the subsequent differences can be large, and differences in likelihood specifications.
- DIC comparisons are most straightforward when the form of the likelihoods (focus) are the same and only the selected explanatory variables differ.
- Specification of the DIC implies use of the data twice: once to produce the posterior and once again with $p(\mathbf{y}|\boldsymbol{\theta})$ in the expectation.
- Calculating DIC values is very difficult when $D(\boldsymbol{\theta})$ cannot be produced in closed form and numerical techniques are necessary.

Returning to the Hiearchical Mode of Lobbying Influence in the US States

```
# CHECK CONVERGENCE
sink("asap2.diags")
asap.mat <- asap.out3$BUGSoutput$sims.array[,1,] # USE THE FIRST OF 3 PARALLEL CHAINS
asap.mcmc <- mcmc(asap.mat)  # TURN IT INTO AN MCMC OBJECT
superdiag(asap.mcmc)  # NOW RUN SUPERDIAG
sink()

# GET THE DEVIANCE AND THE DIC
asap2.dic <- dic.samples(asap2.model, n.iter=2500, type="pD")
Mean deviance:  3861
penalty 34.2
Penalized deviance: 3895
```

- This is a relative measure and needs to be checked against a null model.

Using rjags from R, Repeat Steps for Null Model

```
asap.null.rjags  <- function()  {
for (i in 1:SUBJECTS) {
    nu[i] <- alpha[state.id[i]]
      + beta[1]*contracting[i] + beta[2]*gov.influence[i] + beta[3]*leg.influence[i]
      + beta[4]*elect.board[i] + beta[5]*years.tenure[i]  + beta[6]*education[i]
      + beta[7]*party.ID[i]    + beta[8]*category2[i]     + beta[9]*category3[i]
      + beta[10]*category4[i]  + beta[11]*category5[i]    + beta[12]*category6[i]
      + beta[13]*category7[i]  + beta[14]*category8[i]    + beta[15]*category9[i]
      + beta[16]*category10[i] + beta[17]*category11[i]   + beta[18]*category12[i]
      + beta[19]*med.time[i]   + beta[20]*medt.contr[i]
    mu[i] <- alpha[state.id[i]]
    grp.influence[i] ~ dnorm(mu[i],tau.y)
    for (j in 1:STATES) {
      eta[j] <- gamma[1]*gov.ideology[j]+gamma[2]*lobbyists[j]+gamma[3]*nonprofits[j]
      alpha[j] ~ dnorm(0,tau.alpha)
    }
    :
}
```

Using rjags from R (same model)

```
# SAVE MODEL TO A FILE
write.model(asap.null.rjags, "Article.JPART/asap.null.rjags")

# RUN THE SAMPLER AND COLLECT coda SAMPLES
asap.null.out <- jags(data=asap.jags.list, inits=asap.inits, asap.params, n.iter=5000,
    model="Article.JPART/asap.null.rjags", DIC=TRUE)

# USE THE autojags() FUNCTION TO RUN UNTIL G&R < 1.1
asap.null.out2 <- autojags(asap.null.out)

# OR update() TO RUN LONGER
asap.null.out3 <- update(asap.null.out, n.iter=200000)
```

```
# CHECK CONVERGENCE
superdiag(asap.null.mcmc3)
asap.mat <- asap.null.out3$BUGSoutput$sims.array[,1,]
asap.mcmc <- mcmc(asap.mat)
superdiag(asap.mcmc)

# GET THE DEVIANCE AND THE DIC
asap.null.dic <- dic.samples(asap.null.model, n.iter=2500, type="pD")
```

Deviance Comparison

- Three models:

Model	Deviance	Difference	DF	tail value
Null	3963.8	103.7	24	6.9787e-12
Estimated	3861.1			
Saturated	0.0	3861.1	689	<1.0e-300

- This indicates that our model is statistically distinct from both the null model and the saturated model, meaning that we have substantial progress away from the simplest modeling approach but we still have unexplained variance relative to a fully saturated specification.

Ordered Logit Example

- The 1960 U.S. presidential election between John Kennedy and Richard Nixon was one of the closest contests in national history, with Kennedy's margin of victory less than one percent of the popular vote.
- Campaigns, journalistic accounts, and social contexts can modify pre-election perceptions of the competitiveness of an uncertain outcome.
- This has implications for turnout and therefore may affect the election as well.
- This example uses the 1960 American National Election Study to explore the link between personal characteristics and perception of closeness of the impending election.
- The outcome variable has four ordered categories:
  - ▷ one candidate will win by a lot,
  - ▷ one candidate will win by quite a bit,
  - ▷ this will be a close race–fairly even,
  - ▷ this will be a very close race.

Ordered Logit Example

- The specified explanatory variables are:
  - `education`, 1=8th grade or lower (233 cases), 2=highschool (428 cases), 3=some college or more (185 cases).
  - `sex`, 1=male, (412 cases), 2=female (434 cases).
  - `seedebates`, 1=no (141 cases), 2=yes (660 cases).
  - `importance`, 1=care very much (270 cases), 2=care pretty much (308 cases), 3=pro-con/depends (6 cases), 4=don't care very much (155 cases), 5=don't care at all (85 cases).
  - `involvement`, 8 categories from low to high with the distribution of cases: $(158, 165, 245, 101, 78, 55, 4, 39)$.
  - `catholic`, 1=no (623 cases), 2=yes (179 cases).
  - `partyid`, 1=strong Democrat (198 cases), 2=not very strong Democrat (201 cases), 3=independent closer to Democrats (52 cases), 4=independent (67 cases), 5=independent closer to Republicans (59 cases), 6=not very strong Republican (117 cases), 7=strong Republican (139 cases).
- Note that there are no 0 codings.

Ordered Logit Example

- Ordered choice models are constructed by assuming that there is a continuous latent metric dictating the categorical choices since researchers construct the scale not the respondents.
- So the outcome variable $\mathbf{Y}$ has C ordered categories separated by estimated thresholds (sometimes called "cutpoints" or "fences") sitting over a continuous utility metric $\mathbf{U}$ that cannot be seen:

$$\mathbf{U}_i: \ \theta_0 \underset{c=1}{\Longleftrightarrow} \theta_1 \underset{c=2}{\Longleftrightarrow} \theta_2 \underset{c=3}{\Longleftrightarrow} \theta_3 \ldots \theta_{C-1} \underset{c=C}{\Longleftrightarrow} \theta_C,$$

  where the end-categories extend out to $-\infty$ and ∞, respectively.
- The effect of the explanatory variables is determined by a linear additive specification on the latent scale such that the ith person's utility is $U_i = \mathbf{X}_i\boldsymbol{\beta} + \epsilon_i$, where the $\boldsymbol{\beta}$ do not depend on the θ values.
- Some authors prefer a minus sign in front of $\mathbf{X}_i\boldsymbol{\beta}$, but the model defined here does not as is also the case with the `R` function `polr`.

Ordered Logit Example

- The vector of utilities across individuals is determined in the following way:
  - the probability of the ith person choosing the kth category or less is $p(Y_i \leq k|\mathbf{X})$,
  - this is equal to the probability that the ith person's utility is less than or equal to next threshold to the right of this category, $p(U_i \leq \theta_k)$,
  - now substitute in the linear additive component for the utility to get $p(\mathbf{X}_i\boldsymbol{\beta} + \epsilon_i \leq \theta_k)$,
  - rearrange to leave the stochastic component alone on the left $p(\epsilon_i \leq \theta_k - \mathbf{X}_i\boldsymbol{\beta})$,
  - notice that this is just the CDF of ϵ_i, $F_{\epsilon_i}(\theta_k - \mathbf{X}_i\boldsymbol{\beta})$, at the point $\theta_k - \mathbf{X}_i\boldsymbol{\beta}$,
  - specifying a logistic distribution for this distribution results in the model $p(Y_i \leq k|\mathbf{X}) = [1 + \exp(\mathbf{X}_i\boldsymbol{\beta} - \theta_k)]^{-1}$.
- An ordered probit model can also be created by specifying $p(Y_i \leq k|\mathbf{X}) = \Phi(\theta_k - \mathbf{X}_i\boldsymbol{\beta})$ instead.
- Concerning R: $logitP(Y <= k|x) = zeta_k - eta$ from the help file ($\eta = \mathbf{X}\boldsymbol{\beta}$).

Ordered Logit Example

- Specify two important datastructures:
  - `Q[i,j]`, the cumulative probability that the ith case is in the jth category or less.
  - `p[i,j]`, the marginal probability that the ith case is in the jth category.
- The logistic specification is specified by `for (j in 1:(Ncat-1)) logit(Q[i,j]) <- cut[j] - mu[i]` which loops through each of the first $k-1$ categories relating the linear additive component to a cumulative probability.
- The last cumulative category is not necessary to calculate since it equals one.
- Next we want to calculate the marginal probabilities from the cumulative probabilities.
- The first category is easy since they are equivalent: `p[i,1] <- Q[i,1]`.
- Then we loop through all of the higher categories except for the last, differencing the adjacent cumulative probabilities to get the marginal probabilities: `for (j in 2:(Ncat-1)) p[i,j] <- Q[i,j] - Q[i,(j-1)]` .
- Finally the right-most category is obtained by `p[i,Ncat] <- 1 - Q[i,(Ncat-1)]`.

Ordered Logit Example

- The outcome variable is then modeled with these probabilities with: `close[i] ~ dcat(p[i,1:Ncat])`.
- The last line in the `1:N` loop creates a residual vector, which is then summarized outside of the loop with a standard error function.
- Priors:
  - β coefficients and the cutpoints are given Cauchy priors with 5 degrees of freedom, centered at zero.
  - The last line creates the variable `cut` from the intermediate variable `cut0` to create a sorted form from the prior distribution.

Ordered Logit Example, Code

```
model {
    for (i in 1:Nsub) {
        mu[i] <- beta[1]*education[i] + beta[2]*sex[i]
            + beta[3]*seedebates[i] + beta[4]*importance[i]
            + beta[5]*involvement[i]
            + beta[6]*importance[i]*involvement[i]
            + beta[7]*catholic[i] + beta[8]*partyid[i]
        for (j in 1:(Ncat-1)) { logit(Q[i,j]) <- cut[j] - mu[i] }
        p[i,1] <- Q[i,1]
        for (j in 2:(Ncat-1)) { p[i,j] <- Q[i,j] - Q[i,(j-1)] }
        p[i,Ncat] <- 1 - Q[i,(Ncat-1)]
        close[i] ~ dcat(p[i,1:Ncat])
        E.y[i] <- close[i] - mu[i]
    }
    sd.y <- sd(E.y[])
    for (k in 1:Nvar) { beta[k] ~  dt(0,1,5) }
    for (k in 1:(Ncat-1)) {  cut0[k] ~  dt(0,1,5) }
    cut[1:(Ncat-1)] <- sort(cut0)
}
```

Ordered Logit Example, Results

Parameters	Mean	Std. Error	95% HPD Interval
`education`	0.2773	0.1067	[0.0681: 0.4864]
`sex`	0.3708	0.1410	[0.0944: 0.6472]
`seedebates`	0.4438	0.1813	[0.0884: 0.7992]
`importance`	-0.0432	0.1235	[-0.2852: 0.1988]
`involvement`	-0.3021	0.1045	[-0.5070:-0.0973]
`importance`×`involvement`	0.0534	0.0280	[-0.0015: 0.1084]
`catholic`	0.2484	0.1741	[-0.0928: 0.5896]
`partyid`	0.0403	0.0328	[-0.0240: 0.1047]
θ_1	-1.7139	0.5141	[-2.7215:-0.7064]
θ_2	-0.2879	0.4989	[-1.2658: 0.6900]
θ_2	3.0812	0.5142	[2.0734: 4.0890]

$s_y = 0.7517$, Model DIC: 1580.64, Null DIC: 1609.95

What Is the *Effective Sample Size* In MCMC?

- Notice that the `JAGS` output gave `n.eff` for each parameter.
- This is an estimate of the equivalent number of *independent* samples for that parameter from the chain.
- First note that the autocorrelation function (ACF) or a series of length n, and lag k is the sum of $n-k$ correlations according to:

$$\rho_k = \sum_{i=1}^{n-k}(x_i - \bar{x})(x_{i+k} - \bar{x}) / \sum_{i=1}^{n}(x_i - \bar{x})^2.$$

- ESS is then calculated by

$$\texttt{n.eff} = \frac{M}{1 + 2\sum_{k=1}^{\infty}\rho_k},$$

  where M is the length of the MCMC run.
- ESS affects the Monte Carlo standard error: how much sampling error is due to MCMC rather than independent Monte Carlo samples: `TimeseriesSE` in the `JAGS` output.

Some Common Questions:

- How long should I run the chain before I can claim that it has converged to its invariant (stationary) distribution?
  Preconvergence values are useless.
- How long do I need to run the chain in stationarity before it has sufficiently mixed throughout the target distribution?
  Poor mixing gives biased posterior inferences.
- What does it *mean* if the chain has converged in only some dimensions?
  This is a common but unaddressed occurrence.

Comments on Burn-in

- The burn-in period is the initial time that is considered to be pre-stationarity.
- We run the chain for some time after the starting point and throw away the values.
- Convergence assessment is essential (diagnostics to come).
- It pays to be conservative in deciding the length of the burn-in period.
- There is no "golden rule" here.
- There are now many users of `WinBUGS`, but apparently some of them pay little attention to issues of convergence beyond mechanical testing.
- Since MCMC is specifically used to produce marginal distributions, it is predictable that practitioners would only worry marginally.
- There is a substantial gulf between the theoretical work and practical guidance.

MCMC Convergence

- Empirical summaries from a given MCMC analysis are not reliable until the chain has reached its stationary distribution and had time to sufficiently mix throughout.
- Until X_t converges at time t ($X \sim \pi$), it is not possible to rely upon the effect of any variant of the CLT.
- Therefore it is necessary to convince yourself and your readers that the Markov chain has mixed throughout its stationary distribution.

MCMC Convergence (cont.)

- Three primary problems:
  - There is no *general* proof of convergence for the Gibbs sampler or the more broad Metropolis-Hastings algorithm.
  - There is no way to guarantee that a Markov chain will explore all of the areas of the target distribution in finite time.
  - For a given Markov chain at a given time t, there is no absolute assurance that the chain is currently in the target distribution.
- And Gelman (1996) gives three additional mechanical level worries: an inappropriately specified model, errors in programming the Markov chain (stationary distribution of the chain may not be the desired target), and slow convergence. Markov chain has mixed throughout its stationary distribution.

MCMC Convergence (cont.)

- The good news:
  - All ergodic Markov chains are guaranteed to converge asymptotically.
  - It is often quite easy to determine if a given chain has *not* converged.
  - We have lots of tests.

MCMC Convergence (cont.)

- Three basic approaches:
  - assessing the exact theoretical and mathematical properties of particular Markov chains,
  - diagnosing summary statistics from in-progress models,
  - avoiding the issue altogether with perfect sampling.

MCMC Convergence (cont.)

- Convergence analysis using mathematical properties of the chain:
  - ▷ involves understanding all mathematical consequences of the kernel.
  - ▷ then it is often possible to put restrictions on data or the chain that assure convergence in some finite and known time period.
  - ▷ Sometimes this isn't so bad: all MCMC algorithms converge at a rate related to how close the second largest eigenvalue of the transition matrix (discrete state spaces) or kernel density (continuous state spaces) is to one.
  - ▷ Disadvantages:
    - frequently very complex mathematically,
    - bounds on convergence time can be weak,
    - some of the subsequent restrictions affect statistical specification,
    - generally model-specific.

MCMC Convergence (cont.)

- Convergence monitoring:
  - Stop the chain intermittently and apply empirical diagnostics for nonconvergence.
  - A somewhat subjective, ad-hoc process.
  - **Important principle**: these are indicators of *nonconvergence*. So failing to find evidence of nonconvergence with these procedures is not evidence of convergence.
  - Careful practitioners will use more than one tool here.
  - The `R` package `superdiag` has the function `superdiag` that runs all of commonly used diagnostic tests that we will discuss (and one more).

Example: A Normal-Hierarchical Model of Cold War Military Personnel

- Proportional changes in military personnel for nine east bloc countries (Warsaw Pact plus two) during the period from 1948 to 1983, an interval that covers the height of the Cold War.
- These are collected by Faber (1989, ICPSR-9273) for 78 countries total and include covariates for military, social, and economic conditions.
- The data. . . ...

	Proportional Change in Military Personnel								
	Yugo.	Alb.	Bulg.	Czec.	GDR	Hung.	Poland	Rum.	USSR
1949	0.000	0.083	0.166	0.000	1.000	0.571	0.250	0.006	0.241
1950	0.000	-0.077	0.142	0.000	0.833	0.909	0.286	0.305	0.194
1951	0.498	-0.083	-0.043	0.000	0.864	0.476	0.220	0.234	0.163
1952	0.000	0.109	-0.050	0.000	0.244	0.097	0.125	0.004	0.160
1953	0.000	-0.131	-0.016	0.000	0.255	0.065	0.000	0.004	0.000
1954	0.000	-0.226	0.139	0.000	0.266	0.066	-0.333	0.004	0.000
1955	0.008	-0.049	0.000	0.000	0.296	0.057	0.000	0.000	0.000
1956	0.000	-0.051	-0.103	0.000	0.038	0.054	0.000	-0.143	-0.121
1957	0.000	-0.108	0.005	0.000	0.037	-0.977	0.000	-0.167	-0.118
1958	0.000	-0.061	-0.130	0.000	-0.124	5.000	0.000	0.057	-0.133
1959	0.000	-0.129	-0.144	0.212	-0.131	0.833	-0.333	0.054	-0.077
1960	0.000	0.000	-0.175	0.000	0.291	0.455	0.000	0.051	0.000
1961	0.032	0.037	0.017	-0.182	-0.099	0.438	0.275	-0.022	-0.167
1962	-0.102	0.071	0.125	-0.254	-0.150	0.043	0.008	0.369	0.200
1963	-0.114	0.167	0.096	0.000	0.365	0.125	0.000	-0.056	-0.083
1964	-0.089	0.086	0.115	0.270	0.034	0.030	0.058	-0.017	-0.039
1965	-0.108	0.000	0.012	0.000	0.017	0.036	0.018	-0.085	-0.076
1966	0.146	0.368	0.024	0.085	0.574	0.000	0.173	-0.027	0.159
1967	-0.085	-0.019	0.000	0.039	0.026	-0.049	-0.031	-0.112	0.022
1968	-0.077	0.000	0.012	0.000	-0.005	0.000	0.013	0.000	0.000
1969	-0.008	-0.020	-0.012	0.000	-0.036	-0.058	-0.031	-0.135	0.023
1970	0.084	0.080	-0.029	-0.234	0.069	0.062	-0.071	-0.062	-0.004
1971	-0.105	-0.222	-0.096	-0.064	-0.356	-0.270	-0.059	-0.116	-0.044
1972	0.000	-0.167	0.000	0.000	0.000	0.000	0.000	0.125	0.000
1973	0.043	0.086	0.000	0.000	0.000	0.000	0.037	-0.056	0.015
1974	-0.042	0.000	0.000	0.053	0.154	0.000	0.071	0.000	0.029
1975	0.000	0.000	0.000	0.000	-0.067	0.100	0.000	0.000	0.014
1976	0.087	0.237	0.133	-0.100	0.143	-0.091	0.000	0.059	0.020
1977	0.040	-0.043	-0.118	0.000	0.000	0.000	0.033	0.000	0.008
1978	0.038	-0.089	0.000	0.056	0.000	0.100	0.000	0.000	-0.011
1979	-0.037	0.049	0.000	0.000	0.000	-0.091	0.032	0.000	0.005
1980	0.000	-0.047	0.000	0.053	0.013	-0.070	0.000	0.028	-0.025
1981	-0.027	0.049	-0.007	-0.030	0.031	0.086	0.000	0.000	0.028
1982	-0.008	0.000	-0.007	0.015	-0.006	0.050	0.000	-0.022	0.011
1983	-0.044	-0.070	0.095	0.041	0.006	-0.009	0.063	0.050	-0.043

Example: A Normal-Hierarchical Model of Cold War Military Personnel (cont).

- We want to recognize national differences from economic and cultural factors, while still modeling the political influence that comes from Warsaw Pact membership.
- Data: Y_{ij} are proportional changes in military personnel for country i at time period j and X_j is the index of the year.
- Specification:

$$
\begin{aligned}
Y_{ij} &\sim \mathcal{N}(\alpha_i + \beta_i X_j, \tau_c) \\
\alpha_i &\sim \mathcal{N}(\alpha_\mu, \alpha_\tau) \\
\beta_i &\sim \mathcal{N}(\beta_\mu, \beta_\tau) \\
\alpha_\mu &\sim \mathcal{N}(1, 0.001) \qquad\qquad \alpha_\tau \sim \mathcal{G}(0.001, 0.001) \\
\beta_\mu &\sim \mathcal{N}(0, 0.001) \qquad\qquad \beta_\tau \sim \mathcal{G}(0.001, 0.001) \\
\tau_c &\sim \mathcal{G}(0.001, 0.001),
\end{aligned}
$$

Example: A Normal-Hierarchical Model of Cold War Military Personnel (cont).

- 1000 iterations of the Markov chain are run and disposed of. The monitoring is turned on for all five nodes and then an additional 100,000 iterations are run.
- Results:

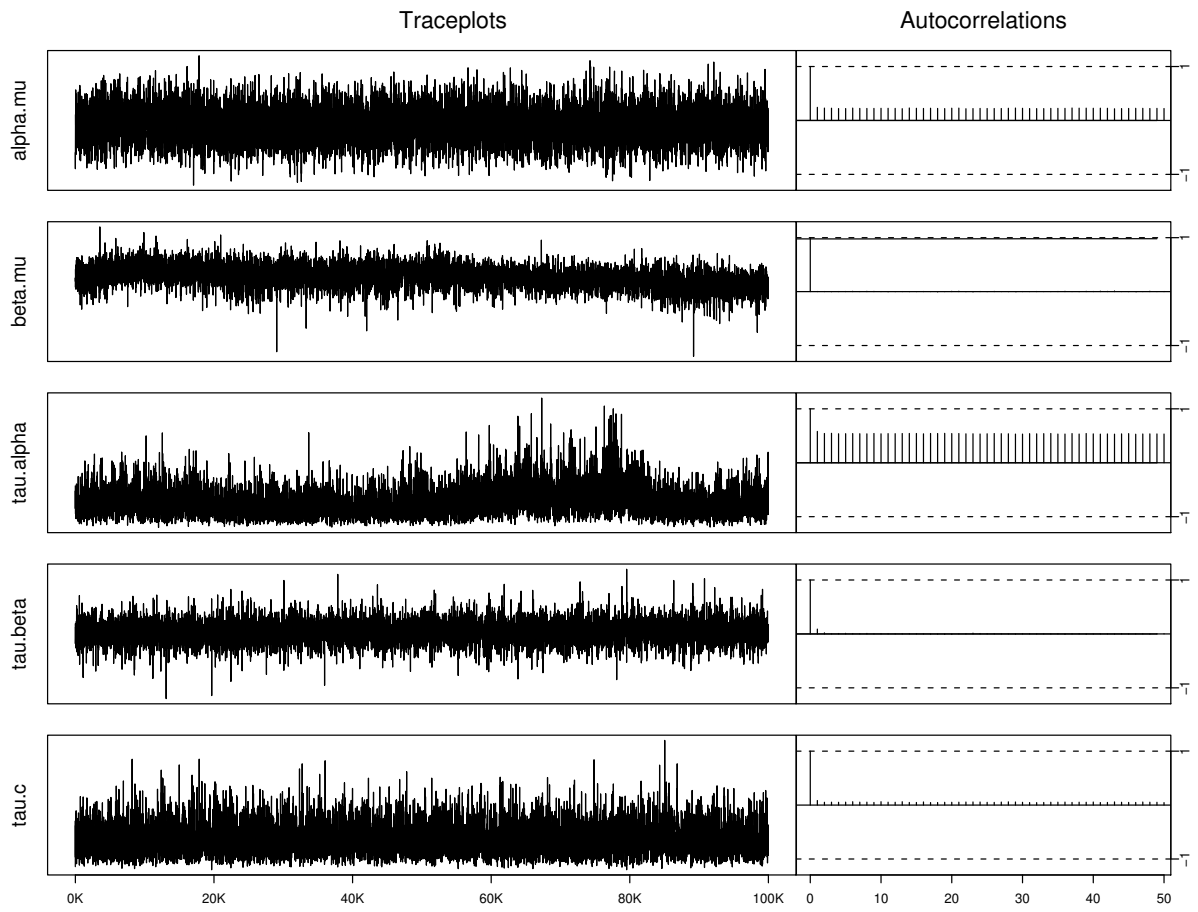
Table 2: Posterior Summary, Military Personnel Model

	Mean	Standard Error	Median	95% HPD
α_μ	9.5502	3.7934	9.7829	[1.4639: 16.4363]
β_μ	-0.0053	0.0192	-0.0053	[-0.0439: 0.0330]
α_τ	0.0150	0.0131	0.0108	[0.0026: 0.0511]
β_τ	398.4455	199.3431	364.7520	[109.3254:875.0422]
τ_c	8.1753	0.6665	8.1613	[6.9151: 9.5301]

Correlation and Autocorrelation

- High correlation *between* the parameters of a chain tends to give slow convergence.
- Whereas high correlation within a single parameter (autocorrelation) chain leads to slow mixing and possibly individual *nonconvergence* to the limiting distribution.

Traceplots and Autocorrelations



Correlation and Autocorrelation (cont.)

- Notes:
  - Traceplots are more informative if the burn-in period is omitted.
  - Things to look for in traceplots: *trends* and *snaking*.
  - Autocorrelation: or a series of length n, the lag k autocorrelation is the sum of $n-k$ correlations according to:

$$\rho_k = \sum_{i=1}^{n-k}(x_i - \bar{x})(x_{i+k} - \bar{x}) / \sum_{i=1}^{n}(x_i - \bar{x})^2$$

  - Usually its not necessary to look beyond lag 50.
  - Autocorrelation problems can often be helped with reparameterization.

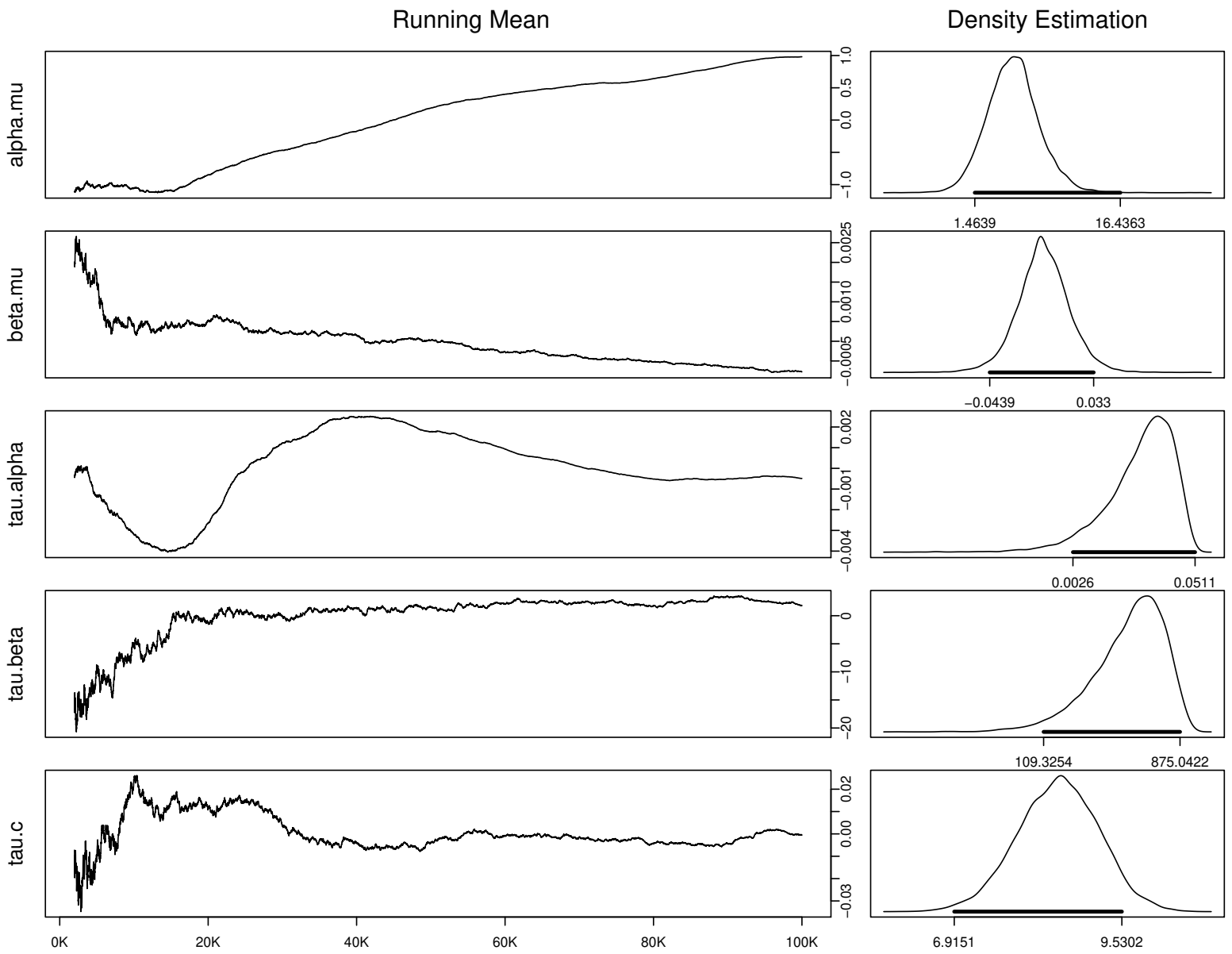
Correlation and Autocorrelation (cont.)

- BOA and CODA give autocorrelation graphs as well as correlation matrices.
- Military Personnel:

	Within-Chain Correlations				Cross-Chain Correlations			
	Lag 1	Lag 5	Lag 10	Lag 50	β_μ	α_τ	β_τ	τ_c
α_μ	**0.2428**	**0.2210**	**0.2251**	**0.2212**	-0.0203	**0.2745**	0.0054	-0.0953
β_μ	0.0030	0.0029	0.0013	0.0023		-0.0131	0.0084	0.0095
α_τ	**0.5813**	**0.5392**	**0.5367**	**0.5327**			0.0005	**-0.1388**
β_τ	0.0896	0.0110	0.0066	0.0039				-0.0077
τ_c	0.0847	0.0544	0.0623	0.0461				

Other Graphical Diagnostics

- Running mean smoother: run after burn-in, look for trends
- Smoothed density estimates: pick some later period. Here last 1000 values for smoother compared with HPD for full 100,000...


Running Mean
Density Estimation
alpha.mu
beta.mu
tau.alpha
tau.beta
tau.c
1.4639
16.4363
−0.0439
0.033
0.0026
0.0511
109.3254
875.0422
6.9151
9.5302
0K
20K
40K
60K
80K
100K

Geweke Time-Series Diagnostic

- General idea: test based on comparing some proportion of the early era of the chain after the burn-in period with some nonoverlapping proportion of the late era of the chain.
- This suggest a difference of means test using an asymptotic approximation for the standard error of the difference.
- Values that are atypical of a standard normal distribution provide evidence that the two selected portions of the chain differ reasonably (in the first moment).

Geweke Time-Series Diagnostic (cont.)

- Preselect two nonoverlapping window proportions, one early in the chain and one later in the chain: $\boldsymbol{\theta}_1$ of length n_1 and $\boldsymbol{\theta}_2$ of length n_2 along with some function of interest $g()$.
- Diagnostic is given by:

$$G = \frac{\bar{g}(\boldsymbol{\theta}_1) - \bar{g}(\boldsymbol{\theta}_2)}{\sqrt{\frac{s_1(0)}{n_1} + \frac{s_2(0)}{n_2}}}.$$

and $s_1(0)$ and $s_2(0)$ are standard errors from the symmetric spectral density functions: the uncorrelated contribution from the individual values to the total variance .

Geweke Time-Series Diagnostic (cont.)

- Preselect two nonoverlapping window proportions, one early in the chain and one later in the chain: $\boldsymbol{\theta}_1$ of length n_1 and $\boldsymbol{\theta}_2$ of length n_2 along with some function of interest $g()$.
- Diagnostic is given by:

$$G = \frac{\bar{g}(\boldsymbol{\theta}_1) - \bar{g}(\boldsymbol{\theta}_2)}{\sqrt{\frac{s_1(0)}{n_1} + \frac{s_2(0)}{n_2}}}.$$

  and $s_1(0)$ and $s_2(0)$ are standard errors from the symmetric spectral density functions: the uncorrelated contribution from the individual values to the total variance .
- Typical test is for $g()$ to be the mean:

$$\bar{g}(\theta_1) = \sum_{i=1}^{n_1} g(\theta_i)/n,$$

$$\bar{g}(\theta_2) = \sum_{i=1}^{n_2} g(\theta_i)/n,$$

Geweke Time-Series Diagnostic (cont.)

- Geweke suggests using the ratios $n_1/n = 0.1$ and $n_2/n = 0.5$.
- If these proportions are held fixed as the chain grows in length, then the central limit theorem applies and G converges in distribution to standard normal.
- Geweke's idea is that more can be learned by one very long chain since it will end up exploring areas where humans might not think to send it.
- Unfortunately the window proportions can greatly affect the value of G, and it is therefore important not to only use the defaults: (0.1/0.5).
- Military personnel model (yes, with only the defaults):

	α_μ	β_μ	α_τ	β_τ	τ_c
z-statistic	**5.0004**	-1.6375	**4.4763**	-0.4826	**2.0621**
p-value	0.0001	0.1015	0.0001	0.6294	0.0392

Gelman and Rubin's Multiple Sequence Diagnostic

- An ANOVA type test based on multiple chains.
- Steps:
  1. Run $m \geq 2$ chains of length $[2n]$ from overdispersed starting points, $(1), (2), \ldots, (m)$:

$$\theta_{(1)}^{[0]}, \theta_{(1)}^{[1]}, \ldots, \theta_{(1)}^{[2n-1]}, \theta_{(1)}^{[2n]}$$
$$\theta_{(2)}^{[0]}, \theta_{(2)}^{[1]}, \ldots, \theta_{(2)}^{[2n-1]}, \theta_{(2)}^{[2n]}$$
$$\vdots$$
$$\theta_{(m)}^{[0]}, \theta_{(m)}^{[1]}, \ldots, \theta_{(m)}^{[2n-1]}, \theta_{(m)}^{[2n]},$$

     The starting points should be determined by overdispersing around suspected or known modes. Discard the first n chain iterations.

Gelman and Rubin's Multiple Sequence Diagnostic (cont.)

- Continuing steps:

2. For each parameter of interest calculate the following:
   - **Within chain variance:**

$$W = \frac{1}{m(n-1)} \sum_{j=1}^{m} \sum_{i=1}^{n} (\theta_{(j)}^{[i]} - \bar{\theta}_{(j)})^2$$

   where $\bar{\theta}_{(j)}$ is the mean of the n values for the j^{th} chain.
   - **Between chain variance:**

$$B = \frac{n}{m-1} \sum_{j=1}^{m} (\bar{\theta}_{(j)} - \bar{\bar{\theta}})^2$$

   where $\bar{\bar{\theta}}$ is the grand mean (mean of means since each subchain is of equal length).
   - **Estimated variance:**

$$\widehat{\text{Var}}(\theta) = (1 - 1/n)W + (1/n)B.$$

Gelman and Rubin's Multiple Sequence Diagnostic (cont.)

- Continuing steps:

  3. Calculate the convergence diagnostic, a single scalar value, called the *estimated scale reduction* (or shrink factor):

$$\sqrt{\widehat{R}} = \sqrt{\frac{\widehat{\text{Var}}(\theta)}{W}}.$$

- Values of $\sqrt{\hat{R}}$ near 1 are evidence that the m chains are all operating on the same distribution (in practice values less than roughly 1.1 or 1.2 are acceptable).
- Unless one is very worried about multimodality of the prior, the number of separate chains need only be about 5 to 10.
- It is easy to monitor $\sqrt{\hat{R}}$ as the Markov chain runs and move on to other diagnostics when we are happy.

Gelman and Rubin's Multiple Sequence Diagnostic Notes

- Underlying principle:
  - Before convergence W underestimates total posterior variation in θ because the chains have not fully explored the target distribution.
  - W is therefore based on smaller differences early in the chain.
  - Also before convergence $\widehat{\text{Var}}(\theta)$ overestimates total posterior variance because the starting points are intentionally overdispersed relative to the target.
  - Once the chains have converged, the difference should be incidental since the chains are exploring the same region and are therefore overlapping.

Gelman and Rubin's Multiple Sequence Diagnostic Notes)

- Problems/Challenges:
  - It is not always easy to obtain suitably overdispersed starting points, since determining their position requires some knowledge of the target distribution to begin with.
  - This is especially a problem in higher dimensions.
  - Actually this point is critical since the test relies upon the underdispersed/overdispersed distributional contrast. EM can help!
  - G&R also suggest starting with a mixture of normals centered at known nodes and improving the quality of density estimate with importance sampling.
  - Normal assumptions may not always be supportable (somewhat mitigated by the Brooks-Gelman modification [1998]; accommodating finite-sample variability in the variance calculations for B and W) by calculating $\sqrt{\hat{R}}$ for α trimmed intervals.

The Gelman and Rubin Diagnostic for the Military Personnel Model

- 5 Markov chains run with overdispersed starting points determined by gridding.
- 50,000 values recorded using the convention of using only the second half the chain.
- The Brooks and Gelman correction also gives a multivariate summary, which is 5.5569 here, giving evidence nonconvergence in a single dimension can dominate this omnibus assessment of model quality.

	α_μ	β_μ	α_τ	β_τ	τ_c
$\sqrt{\hat{R}}$	1.9593	1.0120	1.0735	1.0001	1.0131
$\text{Corrected}_{0.500}$	**2.1540**	1.0122	**1.2105**	1.0001	1.0132
$\text{Corrected}_{0.975}$	**3.6935**	1.0333	**1.6166**	1.0003	1.0362

Heidelberger and Welch Diagnostic

- Approach originally applied in OR for individual parameters in single chains.
- Based on a Brownian Bridge (Wiener Process) over the interval $[0, 1]$, in which the starting and stopping points are tied to the endpoints 0 and 1 and the "string" in between varies.
- Like the Geweke diagnostic, it has an inherently time-series orientation and uses the spectral density estimate.

Heidelberger and Welch Diagnostic

- General procedure:

1. Specify a number of iterations to consider N, an accuracy (ϵ), and an alpha level for the test.
2. The null hypothesis is that the chain is currently in the stationary distribution and the test starts with the full set of iterations. If the test rejects the null, the first 10% of the iterations are discarded and the test is run again.
3. This continues until either 50% of the data have been dismissed or the test fails to reject the null with the remaining iterations.
4. If some proportion of the data are found to be consistent with stationarity, then the *halfwidth* analysis part of the diagnostic is performed.

Heidelberger and Welch Diagnostic (cont.)

- Actually a variant of the Kolmogorov-Smirnov nonparametric test for large sample sizes referred to as the Cramér-von Mises test after the form of the test statistic.
- A parameter, s, is defined as the proportion of the continuing sum of the chain, ranging from zero to one.
- We are interested in the difference between the sum of the first sT of the chain (where T is the total length of the chain), and the mean of the complete set of chain values scaled by sT.
- If the chain were in its stationary distribution, then this difference would be negligible. For the disparity between empirical and theoretical CDFs, the Cramér-von Mises test statistic is

$$C_n(F) = \int_X [F_n(x) - F(x)]^2 dF(x),$$

with a predefined rejection region $C_n(F) > c$, established by normal tail values under asymptotic assumptions and the hypothesis that $F_n(x) \neq F(x)$.

Heidelberger and Welch Diagnostic (cont.)

- Some details:
  - $T =$ the total length of the "accepted" chain after the discards.
  - $s \in [0{:}1] =$ the test chain proportion.
  - $T_{\lfloor sT \rfloor} = \sum_{i=1}^{\lfloor sT \rfloor} \theta_i =$the sum of the chain values from one to the integer value just below sT.
  - $\lfloor sT \rfloor \bar{\theta} =$ the chain mean times the integer value just below sT.
  - $s(0) =$ the spectral density of the chain.

Heidelberger and Welch Diagnostic

- Using these quantities, for any given s, we can construct the test statistic:

$$B_T(s) = \frac{T_{\lfloor sT \rfloor} - \lfloor sT \rfloor \bar{\theta}}{\sqrt{Ts(0)}},$$

  which is the Cramér-von Mises test statistic for sums as cumulative values scaled by the spectral density.

- Now $B_T(s)$ can be treated as an approximate Brownian bridge and tested using normal tail values to decide on the 10% discards (where some adjustments are necessary because $s(0)$ is estimated rather than known).

Heidelberger and Welch Diagnostic (cont.)

- Halfwidth part of the test compares two quantities:
  - Using the proportion of the data not discarded, the halfwidth of the $(1-\alpha)\%$ confidence interval is calculated around the sample mean, where the estimated asymptotic standard error is the square root of spectral density divided by the remaining sample size, $s(0)/(T/2)$.
  - A rough estimate of the variance of the total sample mean, given by the sample mean times some accuracy, ϵ, usually defaulted to 0.1.
- If the halfwidth is less than the rough estimate of the variance then the chain is considered to be converged.
- Warning: this second part of the diagnostic is completely dependent on the first to produce a subchain in the true limiting distribution, and if it has settled for some time on a local maxima, then the halfwidth analysis can be wrong.

Heidelberger and Welch Diagnostic for the Military Personnel Model

- Using the standard defaults in BOA ($\epsilon = 0.1$, $\alpha = 0.05$), the H-W test gives the following output:

	Stationarity Test	Keep	Discard	C-von-M	Halfwidth Test	Mean	Halfwidth
alpha.mu	**failed**	20000	30000	55.3424033	passed	10.78567427	0.133312358
beta.mu	**failed**	20000	30000	5.8618976	passed	-0.00570531	0.000238181
tau.alpha	**failed**	20000	30000	16.2614855	passed	0.02185464	0.000795133
tau.beta	passed	50000	0	0.1940956	passed	398.44545696	1.949360870
tau.c	**failed**	20000	30000	44.6781527	passed	8.06357154	0.013199792

CODA and BOA for Post-BUGS Analysis

- CODA ("Convergence Diagnostics and Output Analysis") by Martyn Plummer.
- BOA ("Bayesian Output Analysis") by Brian Smith.
- Both free packages that run under R, and are downloaded at CRAN.
- Both provide a rich set of diagnostic tools, including graphics.
- Both can be a little quirky, but are very useful.

Using CODA

```
> install.packages(pkgs="coda",lib=".Rlib")
> codamenu()
CODA startup menu
1:Read BUGS output files
2:Use an mcmc object
3:Quit
Selection: 1
Enter BUGS output filenames, separated by return key
(leave blank to exit)
1: Article.P-Agent/exec13.4
Abstracting k[1] ... 500000 valid values
Abstracting k[2] ... 500000 valid values
Abstracting k[3] ... 500000 valid values
Abstracting k[4] ... 500000 valid values
Abstracting sigma ... 500000 valid values
Abstracting theta[1] ... 500000 valid values
Abstracting theta[2] ... 500000 valid values
Abstracting theta[3] ... 500000 valid values
```

```
Abstracting theta[4] ... 500000 valid values
Abstracting theta[5] ... 500000 valid values
Abstracting theta[6] ... 500000 valid values
Abstracting theta[7] ... 500000 valid values
Abstracting theta[8] ... 500000 valid values
Abstracting theta[9] ... 500000 valid values
Abstracting theta[10] ... 500000 valid values
Abstracting theta[11] ... 500000 valid values
Abstracting theta[12] ... 500000 valid values
```

Using BOA

```
library(boa)
boa.menu()

Bayesian Output Analysis Program (BOA)
Version 1.0.0 for UNIX R
Copyright (c) 2001 Brian J. Smith <brian-j-smith@uiowa.edu>

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License or any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

For a copy of the GNU General Public License write to the Free
Software Foundation, Inc., 59 Temple Place - Suite 330, Boston,
```

```
MA  02111-1307, USA, or visit their web site at
http://www.gnu.org/copyleft/gpl.html

NOTE: if the menu unexpectedly terminates, type "boa.menu(recover= TRUE)" to
restart and recover your work

BOA MAIN MENU
*************
1:File     >>
2:Data     >>
3:Analysis >>
4:Plot     >>
5:Options  >>
6:Window   >>

FILE MENU
=========
1:Import Data         >>
2:Load Session
3:Save Session
4:Return to Main Menu
```

```
5:Exit BOA
Selection: 1

IMPORT DATA MENU
----------------
1:BUGS Output File
2:Flat ASCII File
3:Data Matrix Object
4:View Format Specifications
5:Options...
6:Back
7:Return to Main Menu
Selection: 1

Enter filename prefix without the .ind or .out extension [Working Directory: ""]
1: Article.P-Agent/exec.short
Read 1 items
Read 18 records

Read 180000 records
+++ Data successfully imported +++
```

```
IMPORT DATA MENU
----------------
1:BUGS Output File
2:Flat ASCII File
3:Data Matrix Object
4:View Format Specifications
5:Options...
6:Back
7:Return to Main Menu
Selection: 7

BOA MAIN MENU
*************
1:File     >>
2:Data     >>
3:Analysis >>
4:Plot     >>
5:Options  >>
6:Window   >>
Selection: 3
```

```
ANALYSIS MENU
============
1:Descriptive Statistics  >>
2:Convergence Diagnostics >>
3:Options...
4:Return to Main Menu
Selection: 1

DESCRIPTIVE STATISTICS MENU
---------------------------
1:Autocorrelations
2:Correlation Matrix
3:Highest Probability Density Intervals
4:Summary Statistics
5:Back
6:Return to Main Menu
Selection: 1
Selection: 3

HIGHEST PROBABILITY DENSITY INTERVALS:
```

```
=====================================

Alpha level = 0.05

Chain: Article.P-Agent/exec.short
---------------------------------

          Lower Bound Upper Bound
k[1]        -10.05000    -0.88130
k[2]         -5.93900     1.43900
k[3]         -1.82500     4.34400
k[4]          4.33500    12.20000
sigma         3.21600     8.75000
tau           0.00945     0.06781
theta[10]     0.22970     1.33000
theta[11]    -1.08300     0.05925
theta[12]     0.62890     3.93200
theta[1]     -4.13900     2.87200
theta[2]     -1.86500     0.55190
theta[3]     -0.83480     0.17190
theta[4]      0.36910     2.05700
```

```
theta[5]      0.28810     3.25600
theta[6]     -3.68100    -1.61800
theta[7]      0.54550     2.15500
theta[8]     -1.66400    -0.35180
theta[9]     -0.14270     0.86110

Selection: 4

SUMMARY STATISTICS:
===================

Bin size for calculating Batch SE and (Lag 1) ACF = 50

Chain: Article.P-Agent/exec.short
-------------------------------

                Mean         SD     Naive SE     MC Error    Batch SE
k[1]      -5.59806203 2.42271566 0.0242271566 0.1223327834 0.166725341
k[2]      -2.42940786 1.94031614 0.0194031614 0.0968573426 0.132938124
k[3]       1.35981060 1.60391981 0.0160391981 0.0778199101 0.108735974
k[4]       8.40854860 1.96700713 0.0196700713 0.1001023043 0.134728021
```

sigma	5.81126050	1.40852457	0.0140852457	0.0750363453	0.097579773
tau	0.03498487	0.01690473	0.0001690473	0.0008895715	0.001164072
theta[10]	0.76229775	0.29844260	0.0029844260	0.0153456628	0.020729837
theta[11]	-0.47943636	0.29944492	0.0029944492	0.0145505332	0.020824001
theta[12]	2.12008306	0.82920162	0.0082920162	0.0403465492	0.049794374
theta[1]	-0.44837512	2.13613478	0.0213613478	0.1104139879	0.150234893
theta[2]	-0.64348071	0.63055720	0.0063055720	0.0309518537	0.038675137
theta[3]	-0.31625976	0.25289936	0.0025289936	0.0131013569	0.017111244
theta[4]	1.11830299	0.44912150	0.0044912150	0.0226997128	0.031597119
theta[5]	1.71032372	0.77381701	0.0077381701	0.0352462425	0.045846571
theta[6]	-2.67945860	0.58558035	0.0058558035	0.0312714378	0.041294397
theta[7]	1.18996756	0.47287189	0.0047287189	0.0219656503	0.033222243
theta[8]	-0.85445191	0.33927737	0.0033927737	0.0181838079	0.023634776
theta[9]	0.34877322	0.24568023	0.0024568023	0.0128376865	0.016650594

	Batch ACF	0.025	0.5	0.975	MinIter	MaxIter	Sample
k[1]	0.9075014	-9.87607500	-5.73300	-0.63469500	1	10000	10000
k[2]	0.8830128	-5.81300000	-2.56200	1.60505000	1	10000	10000
k[3]	0.8511580	-1.88007500	1.40200	4.30900000	1	10000	10000
k[4]	0.8960185	4.43700000	8.49550	12.31000000	1	10000	10000
sigma	0.9565878	3.62395000	5.70100	9.32520000	1	10000	10000

tau	0.9305307	0.01149950	0.03077	0.07616125	1	10000	10000
theta[10]	0.9185856	0.23176000	0.75560	1.33500000	1	10000	10000
theta[11]	0.9200565	-1.03525000	-0.47600	0.16190000	1	10000	10000
theta[12]	0.5477688	0.55450000	2.07700	3.88300000	1	10000	10000
theta[1]	0.9651961	-4.03800000	-0.26025	3.02500000	1	10000	10000
theta[2]	0.6337737	-1.98707500	-0.57785	0.49360000	1	10000	10000
theta[3]	0.8160511	-0.83090000	-0.30690	0.19560000	1	10000	10000
theta[4]	0.9594892	0.41830000	1.04000	2.13605000	1	10000	10000
theta[5]	0.6044240	0.31262750	1.67100	3.31900000	1	10000	10000
theta[6]	0.9783000	-3.80200000	-2.58350	-1.69700000	1	10000	10000
theta[7]	0.9560934	0.52260000	1.09300	2.13900000	1	10000	10000
theta[8]	0.9260070	-1.76300000	-0.77130	-0.38477250	1	10000	10000
theta[9]	0.8242078	-0.08368125	0.32735	0.93360000	1	10000	10000