MCMC: Gibbs Sampling

Edps 590BAY

Carolyn J. Anderson

Department of Educational Psychology

ILLINOIS

Fall 2021

Overview

- Bayesian Computing
- Markov Chain
- Gibbs sampling for univariate Normal
- jags
- Assessing convergence of algorithm
- Simple Linear Regression
- Missing data
- Practice

Depending on the book that you select for this course, read either Gelman et al. pp 275-278 or Kruschke pp 143–221. I am relying on Krucshke and jags documentation for material on jags, and some from Hoff. Also I used the coda, jags, rjags, and runjags manuals.

Introduction to Bayesian Computing

- Our major goal is to approximate the posterior distributions of unknown parameters and use them to estimate parameters.
- The analytic computations are fine for simple problems,
  - Beta-binomial for bounded counts
  - Normal-normal for continuous variables
  - Gamma-Poisson for (unbounded) counts
  - Dirichelt-Multinomial for multicategory variables (i.e., a categorical variable)
  - Models in the exponential family with small number of parameters
- For large number of parameters and more complex models
  - Algebra of analytic solution becomes overwhelming
  - Grid takes too much time.
  - Too difficult for most applications.

Basic Steps in Modeling

Recall that the steps in an analysis:

1. Choose
   - $p(y|\theta)$, the likelihood or data model
   - $p(\theta)$, prior
2. Find $p(\theta|y)$ (posterior) or at least a good approximation of it.
3. Assess convergence
4. Summarize distribution $p(\theta|y)$ or compute desired statistics.

Markov Chain

(quote from Gelman et al.):

"... a Markov Chain is a sequence of random variables $\theta^{(1)}, \theta^{(2)}, \ldots,$ for which, for any t, the distribution of $\theta^{(t)}$ given all previous θ's depends only on the most recent value."
In other words, the future value only depends on the current value but none of the previous ones.

Markov property is

$$\theta^{(t)} \text{ independent of } \theta^{(t-2)}, \theta^{(t-3)}, \ldots$$

The sequence that has the Markov Property $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(S)}$ is a dependent chain or Markov chain.

Markov Chain

(quote from Gelman et al.): "The key to the method's success, however, is not the Markov property but rather that the approximate distributions are improved at each step in the simulation, in the sense of converging to the target distribution."

"The transition probability distributions must be constructed to converge so that the Markov chain converges to a unique stationary distribution that is the posterior distribution, $p(\theta|y)$.

"MCMC"
Markov Chain is a stochastic process
Monte Carlo is a simulation

Overview of MCMC Methods

Common sampling methods

- Gibbs Sampling
- Metropolis
- Metropolis-Hasting algorithm
- Hamiltonian Sampling

Implementations in R

- Program in base R
- Gibbs via jags called from R (e.g., rjags, runjags, jagsUI)
- Hamiltonian via stan called from R using rstan (or brms)

Many useful tools are in the "coda" package, especially for assessing convergence. Also, there are nice functions in Kruschke (2014) text that can be set as source in R. See course-web

Gibbs Sampling

Gibbs works by sampling one parameter at a time from the posterior distribution conditional on other parameters and data.

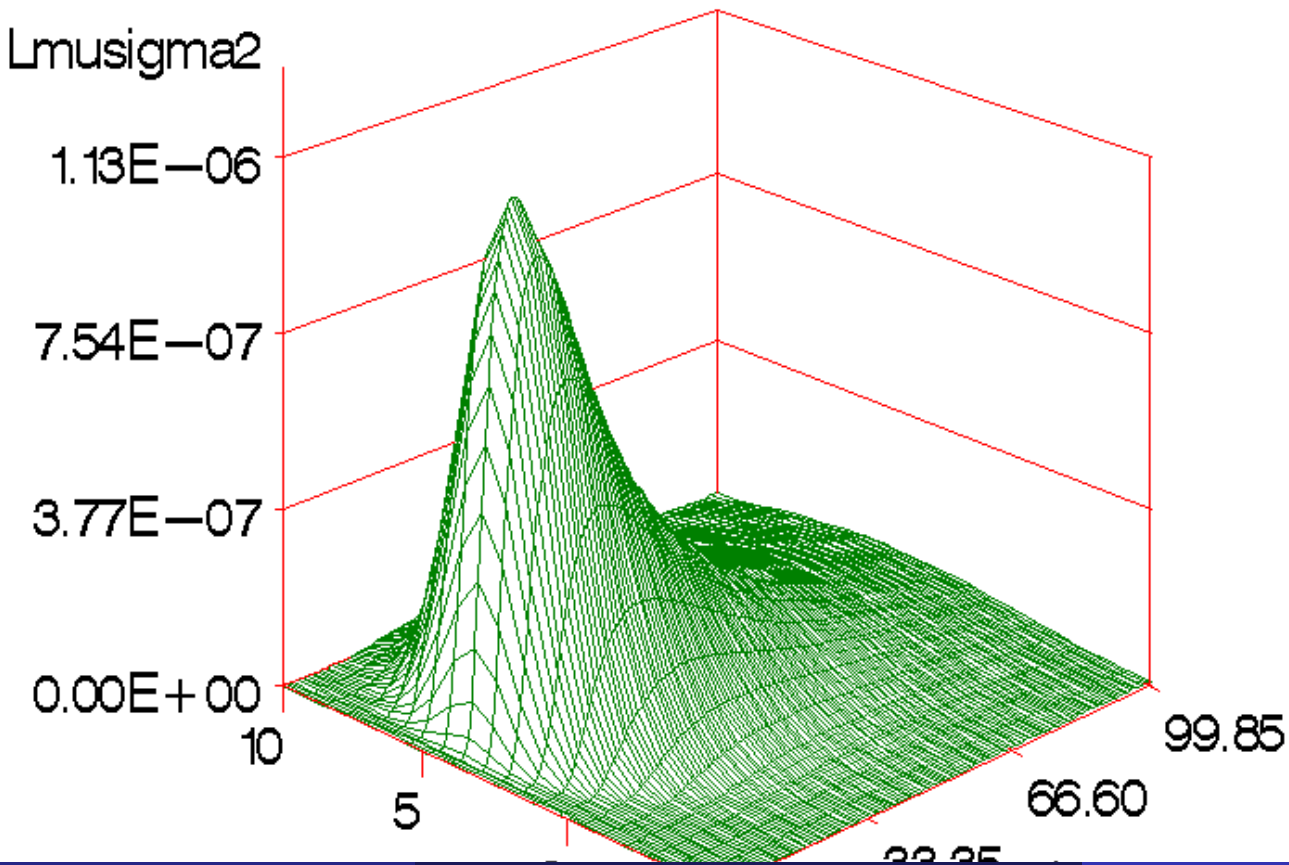
For example, suppose $y_i \sim N(\mu, \sigma^2)$ and the posterior is $p(\mu, \sigma^2|\boldsymbol{y})$. (note: $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$)

1. Draw sample value $\mu^{(t+1)}$ from $p(\mu^{(t)}|\sigma^{(t)}, \boldsymbol{y})$
2. Sample a value of $\sigma^{2(t+1)}$ from $p(\sigma^2|\mu^{(t+1)}, \boldsymbol{y})$

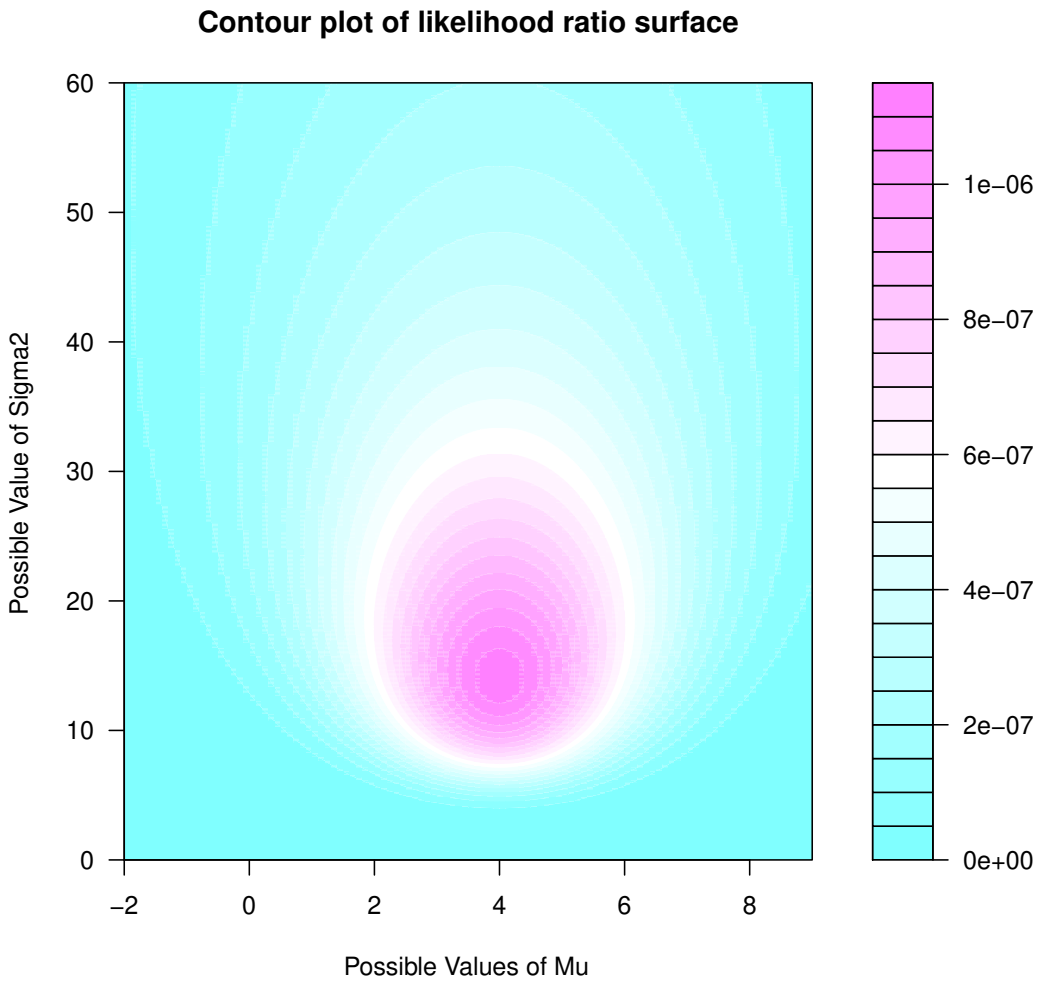
Each time you run steps 1 & 2, you get new values of $\{\mu, \sigma^2\}$ and the repeating these steps many, many times yields an approximation of the posterior distribution, $p(\mu, \sigma|\boldsymbol{y})$.

A Picture of How It Works: μ and σ^2

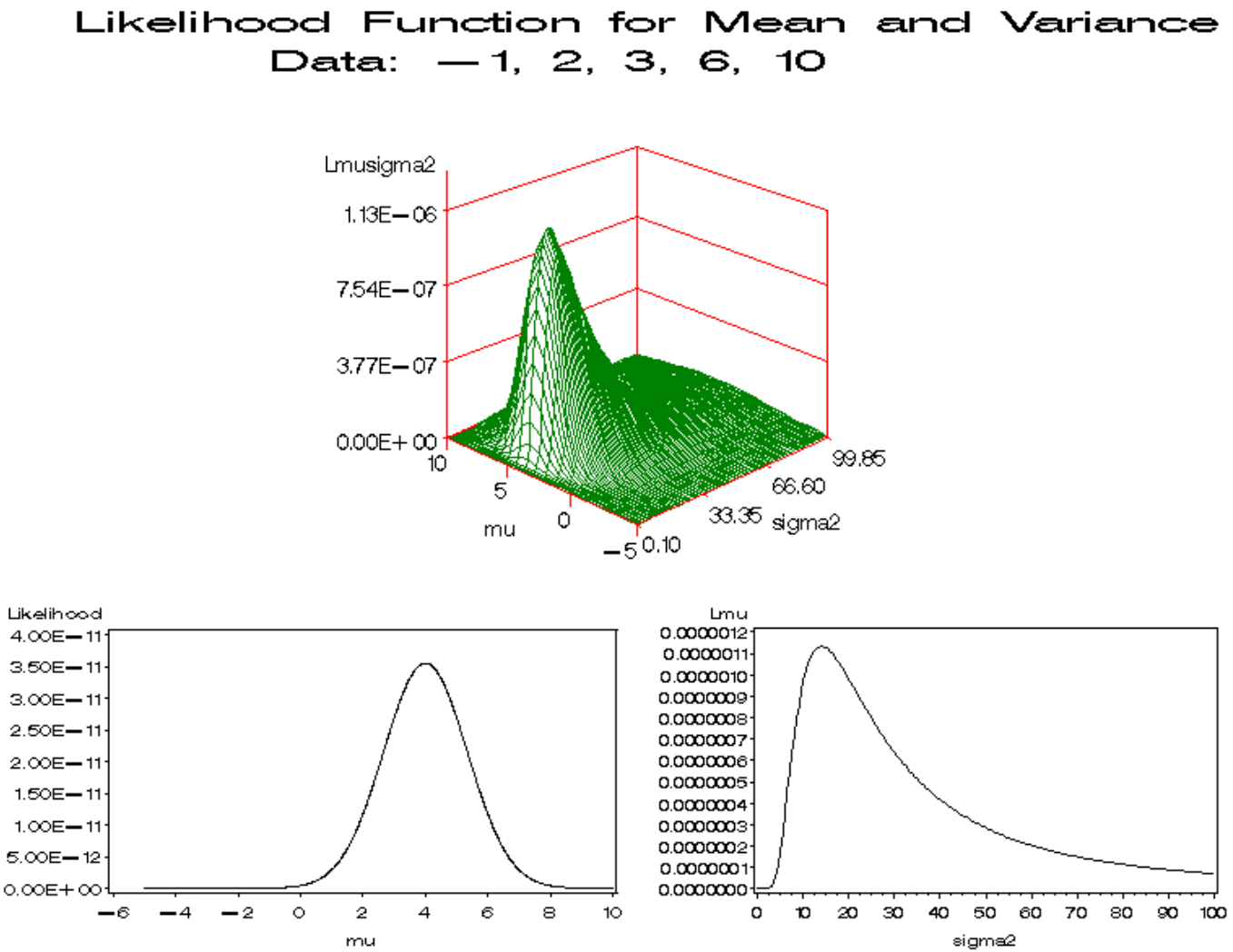
Data: $y_1 = -1, y_2 = 2, y_3 = 3, y_4 = 6, y_5 = 10$



Another View



Univariate Normal: μ and σ^2



Details for Univariate Normal

Suppose the we known μ (or have an estimate of it). The (full) conditional distribution of $\tilde{\sigma}^2$ (remember $\tilde{\sigma}^2 = 1/\sigma^2$, the precision) is by Bayes Theorem

$$\begin{aligned} p(\tilde{\sigma}^2|\mu, y_1, \ldots, y_n) &\propto p(y_1, \ldots, y_n|\mu, \tilde{\sigma}^2)p(\mu, \tilde{\sigma}^2) \\ &\propto p(y_1, \ldots, y_n|\mu, \tilde{\sigma}^2)p(\mu|\tilde{\sigma}^2)p(\tilde{\sigma}^2) \end{aligned}$$

Skipping algebra, $p(\tilde{\sigma}^2|\mu, y_1, \ldots, y_n)$ is gamma; that is,

$$\tilde{\sigma}^2|\mu, y_1, \ldots, y_n \sim \text{ Gamma}(\nu_n/2, \nu_n\sigma_n^2(\mu)/2)$$

Details for Univariate Normal (continued)

$$\tilde{\sigma}^2|\mu, y_1, \ldots, y_n \sim \text{ Gamma}(\nu_n/2, \nu_n\sigma_n^2(\mu)/2)$$

where

$$\begin{aligned}
\nu_n &= \nu_0 + n \\
\sigma_n^2(\mu) &= \frac{1}{\nu_n}\left[\nu_0\sigma_0^2 + ns_n^2(\mu)\right] \\
s_n^2(\mu) &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \mu)^2 \\
ns_n^2(\mu) &= \sum_{i=1}^{n}(y_i - \bar{y} + \bar{y} - \mu)^2 \\
&= \sum_{i=1}^{n}\left((y_i - \bar{y})^2 + 2(y_i - \bar{y})(\bar{y} - \mu) + (\bar{y} - \mu)^2\right) \\
&= (n-1)s^2 + n(\bar{y} - \mu)^2
\end{aligned}$$

And full conditional for θ

$$
\begin{aligned}
\mu|\sigma^2, y_1, \ldots, y_n &\sim N(\mu_n, \tau_n^2) \\
\mu &= \frac{\mu_0/\tau_o^2 + n\bar{y}/\sigma^2}{(\kappa_o/\tau_o^2 + n/\sigma^2)} \\
\tau_n^2 &= \left(\frac{1}{\tau_o^2} + \frac{n}{\sigma^2}\right)^{-1} = \frac{1}{\left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}\right)}
\end{aligned}
$$

We derived these results back in the lecture on "Inference for normal mean and variance".

Little R function call "gibbs" w/ "Semi-Conjugate Priors"

Create some data:

```
mu ← 4
sigma ← 2
n ← 50
y ← rnorm(n,mu,sigma)
```

We also need sample statistics to start things offplo:

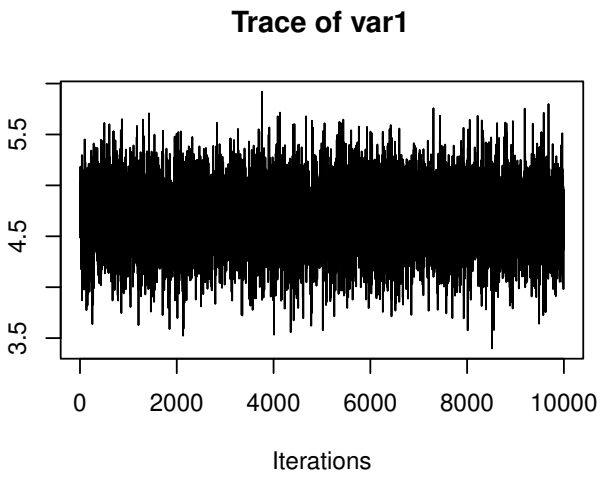
```
ybar ← mean(y)
s2 ← var(y)
```

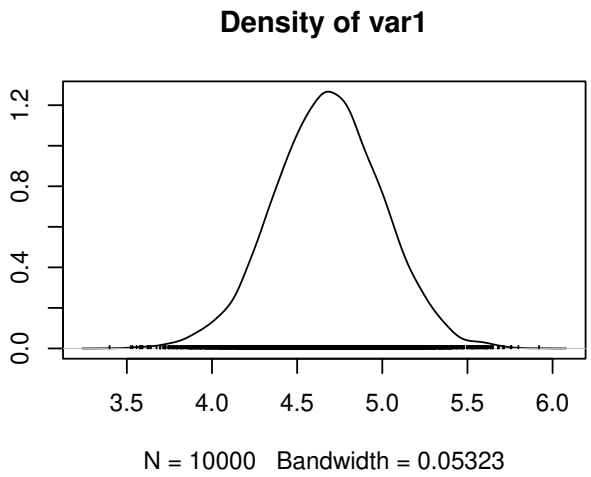
And priors and other things:

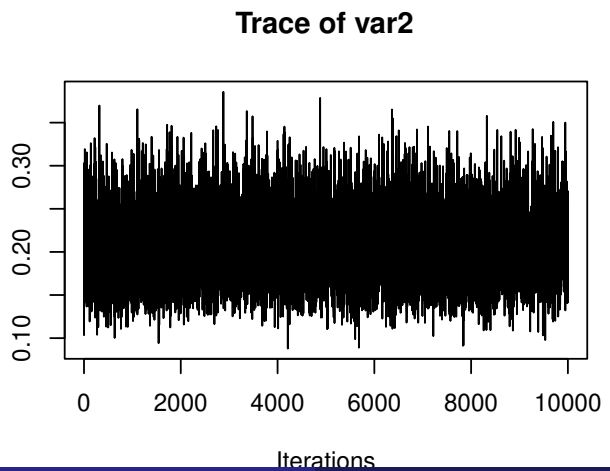
```
mu0 ← 0          t02 ← 100
s02 ← 1          nu0 ← 1
seed ← 2374      S ← 1000
```

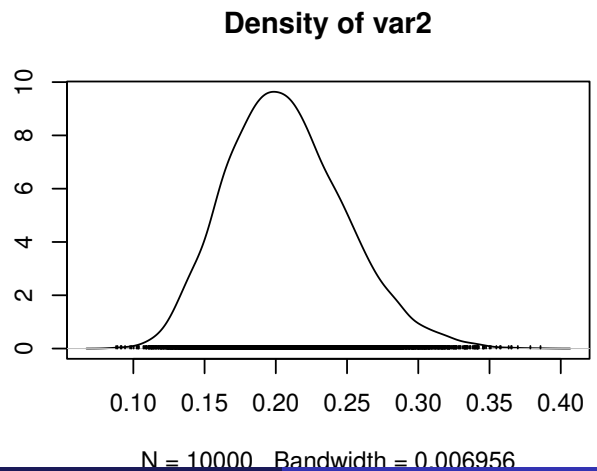
To run it: `sim1 ← gibbs(S,y,mu0,t02,s02,nu0,seed)`

Trace Plots

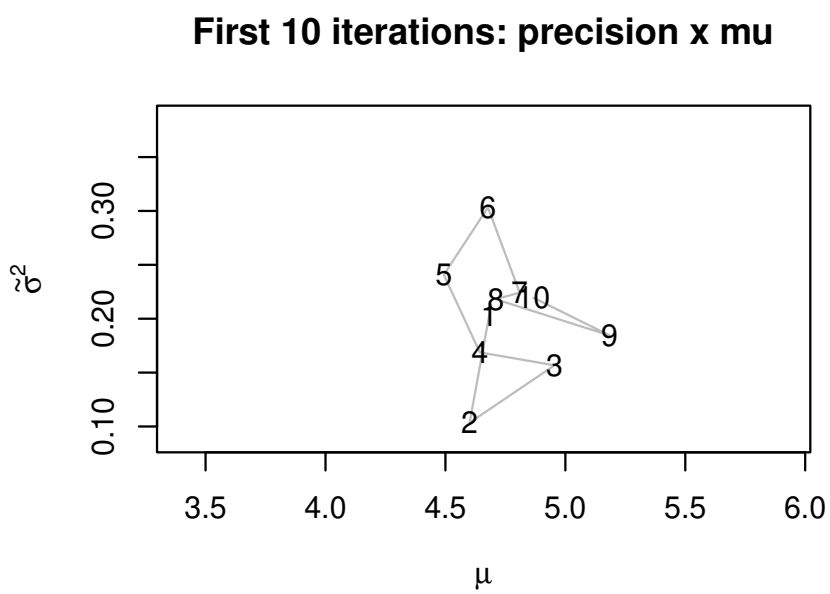

Trace of var1
3.5
4.5
5.5
0
2000
4000
6000
8000
10000
Iterations

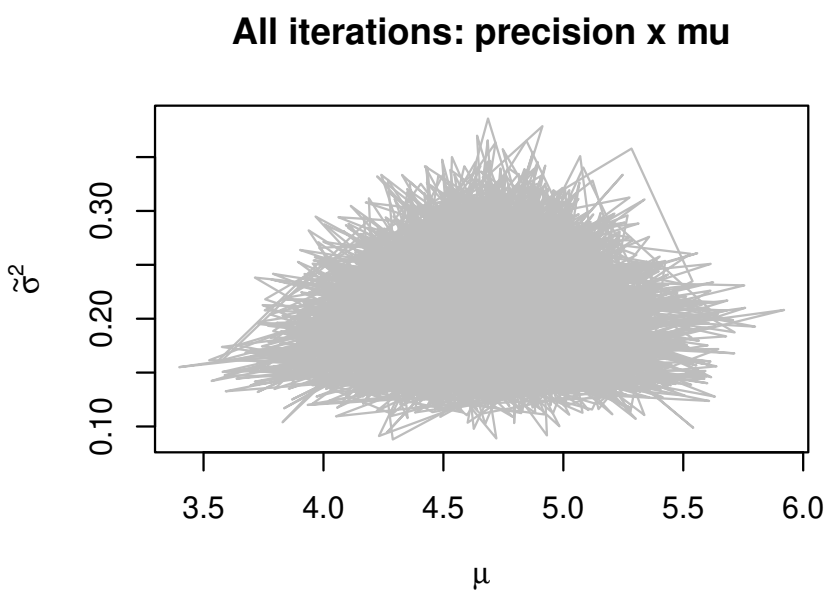

Density of var1
0.0
0.4
0.8
1.2
3.5
4.0
4.5
5.0
5.5
6.0
N = 10000 Bandwidth = 0.05323


Trace of var2
0.10
0.20
0.30
0
2000
4000
6000
8000
10000
Iterations

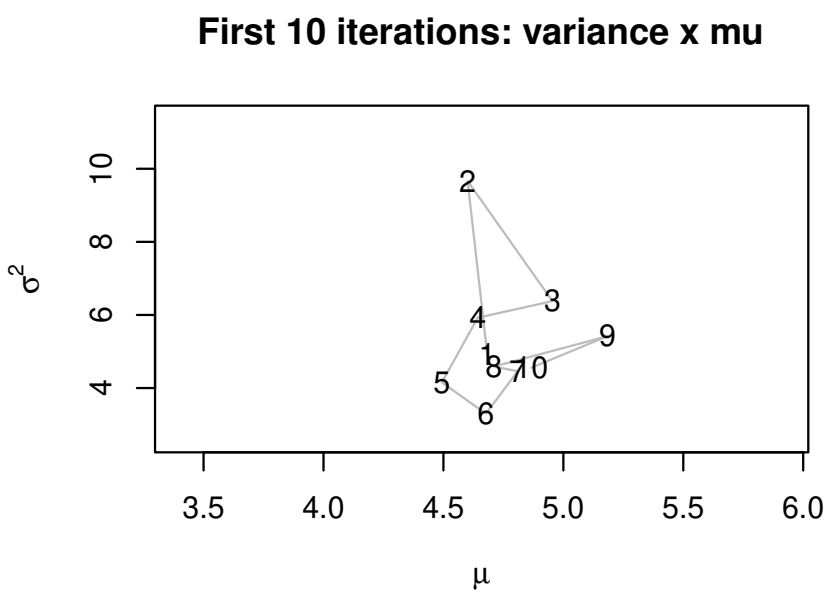

Density of var2
0
2
4
6
8
10
0.10
0.15
0.20
0.25
0.30
0.35
0.40
N = 10000 Bandwidth = 0.006956

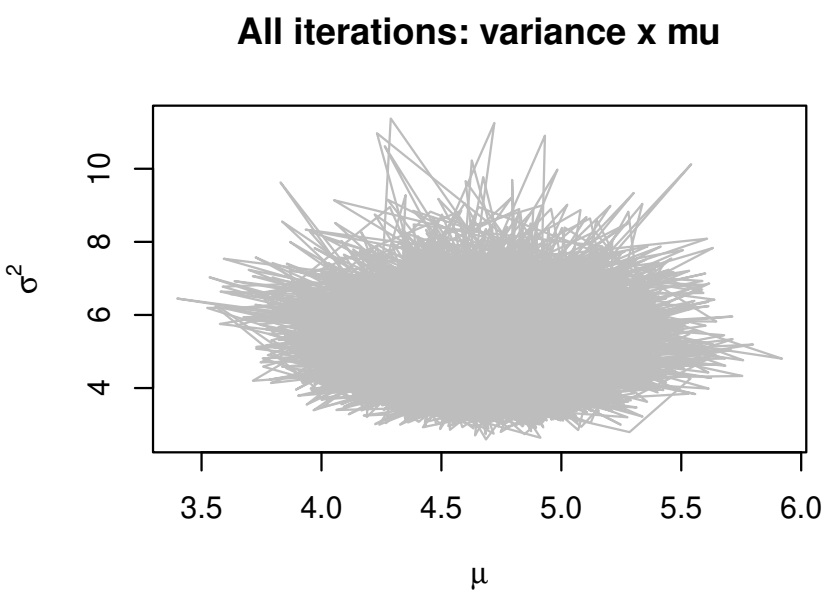
Joint Wandering: Mean & Precision



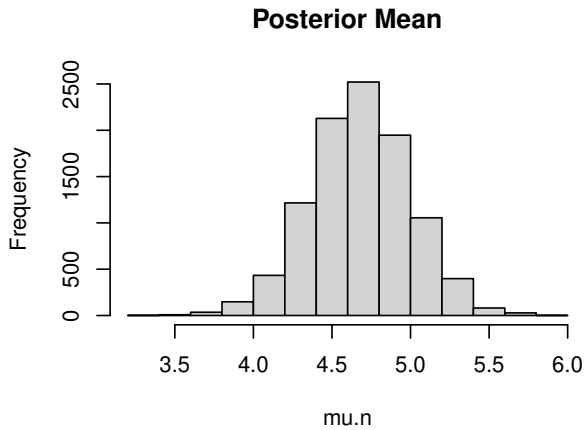


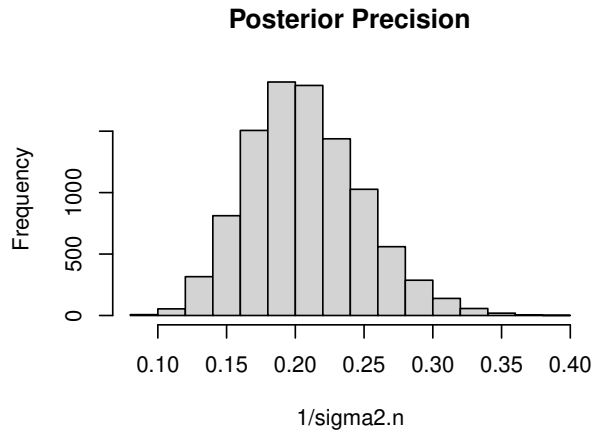
Joint Wandering: Mean & Variance

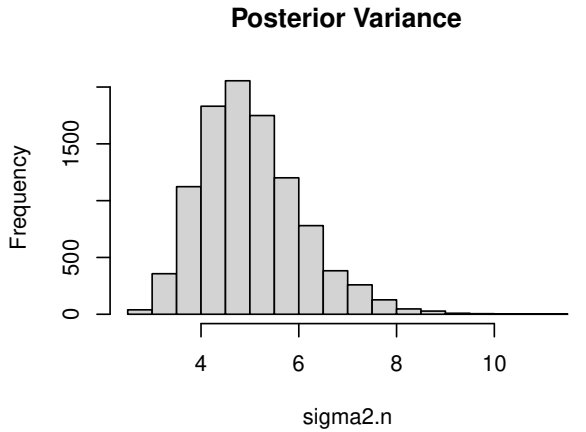


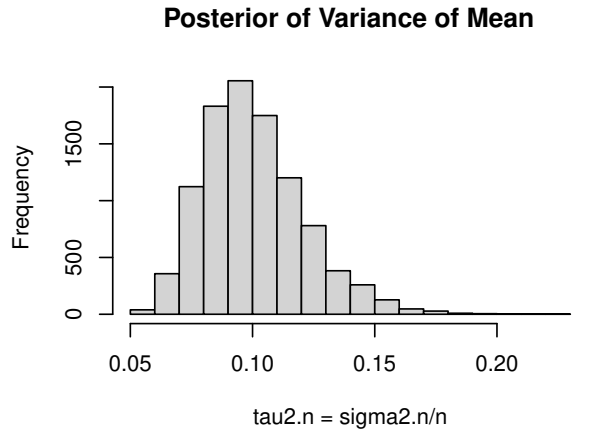


Posterior Distributions









Description of Posterior

Statistics are based on all and the last 50% of the iterations. These are mean values.

Estimated Parameters of the Posterior Distribution

	μ	τ^2	σ^2	$\sqrt{\sigma^2}$
All (10,000)	4.6811	0.1005	5.0272	2.2422
Last 50% (5,000)	4.6821	0.1007	5.0353	2.2439

95% High Density intervals

	All		Last 50%	
Parameter	Lower	Upper	Lower	Upper
μ	4.0662	5.3165	4.0616	5.3143
$1/\sigma^2$	0.1318	0.2893	0.1313	0.2882
σ^2	3.2456	7.1798	3.1796	7.0938

Note: Values may differ from when re-run function

More parameters

It is just the same, but more steps.

Suppose that the parameters that we need to estimate are $\{\theta_1, \theta_2, \ldots, \theta_p\}$. We use the full conditionals and sample for one at a time:

$$p(\theta_i | \theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_p, y_1, \ldots, y_n)$$

1. Sample $\theta_1^{(t+1)}$ from $p(\theta_1 | \theta_2^{(t)}, \theta_3^{(t)}, \ldots, \theta_p^{(t)}, y_1, \ldots, y_n)$
2. Sample $\theta_2^{(t+1)}$ from $p(\theta_2 | \theta_1^{(t+1)}, \theta_3^{(t)}, \ldots, \theta_p^{(t)}, y_1, \ldots, y_n)$
3. $\vdots$
4. Sample $\theta_p^{(t+1)}$ from $p(\theta_p | \theta_1^{(t+1)}, \theta_2^{(t+1)}, \ldots, \theta_{p-1}^{(t+1)}, y_1, \ldots, y_n)$

Then repeat many, many times

Advantage and Disadvantages of Gibbs

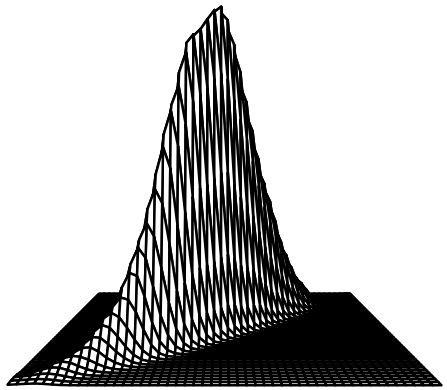
- No tuning needed
- More efficient than Metropolis-Hastings (don't reject any sample values)
- Is a special case of Metropolis-Hastings
- Can run a Metropolis-Hastings for each parameter within Gibbs Sampling

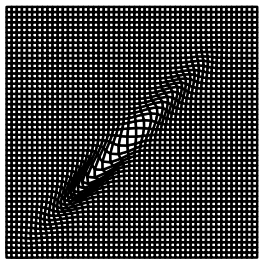
But

- Must be able to derive full conditionals for each parameter.
- Must be able to sample from these full conditionals.
- It can get "stuck" or be very slow when parameters are highly correlated (e.g., intercept and slope in regression)

Sampler Can Get Stuck

https://chi-feng.github.io/mcmc-demo/app.html (circle and Gibbs)
Sampling gets "stuck" along ridge. Below is the density of a bivariate normal with $r \sim .98$.





jags: Implementing Gibbs

We could write functions or R functions and scripts to implement Gibbs (as done in the little function ‘‘`gibbs`’’), but there is an easier way where we only need to input

- Data: `dataList`
- Model: Specify the likelihood and prior
- Starting values: `initsList`

We will discuss all steps in the context of anorexia data for unknown mean and variance

Packages in RI

There are (at least) 4 packages that implement Gibbs via jags in R:

- rjags
- runjags
- jagsUI
- R2jags

We will discuss the first three, but we won't cover all the options at this time.

Steps in Running a Model

1. Set up data
2. Define the model
3. Initialization
4. Adaptation and burn-in or "warm-up"
5. Monitoring
6. Assess convergence
7. Model evaluation/criticism
8. Summarize results (posterior distribution)

Set-up Anorexia Data for jags

```
dataList  ←  list(y=ano$change,
             Ntotal=length(ano$change),
             meanY = mean(ano$change),
             sdY = sd(ano$change)
             )
```

- "dataList" is a list object.
- y is the outcome or response variable (our data)
- meanY and sdY are used in the model specification.

Model for Anorexia Data

```
Model1  =  "model {
            for (i in 1:Ntotal){
              y[i] ~ dnorm( mu, precision)
              }
            mu ~ dnorm( meanY , 1/(100*sdY2) )
            precision <- 1/sigma2
            sigma ~ dunif( sdY/1000, sdY*1000 )
            }
            "


writeLines(Model1, con=''Model1.txt'')
```

- A character object written to hard drive
- dnorm takes as parameters the mean and precision.
- "precision" is precision $= 1/\text{var}(y)$.
- "sigma" $=$ sd(y) $\sim$ uniform(.00798,7983.598).

Starting Values

thetaInit = mean(ano$change)
sigmaInit = sd(ano$change)

initsList = list(mu=thetaInit, sigma=sigmaInit)

For us:

```
> initsList
$mu
2.763889

$sigma
7.983598
```

This is our starting point.

Compile and Initialize the Model

```
jagsModel1 ← jags.model(file="Model1.txt",
                        data=dataList,
                        inits=initsList,
                        n.chains=4,
                        n.adapt=500)
```

- jags.model compiles and initializes the model.
- I requested 4 chains.
- n.adapt indicates that adaption during the burn-in or "warm-up" phase should be done; that is, the program self regulates itself to optimize this step.

Getting Samples and Summary

```
# Gets the samples
update (jagsModel1, n.iter=500)

# Contains samples from all chains with 500
# iterations
Samples ← coda.samples(jagsModel1,
      variable.names=c("mu","precision","sigma"),
      n.iter=4000)

# output summary information
summary(Samples)
plot(Samples)
```

rjags output

See Rmarkdown

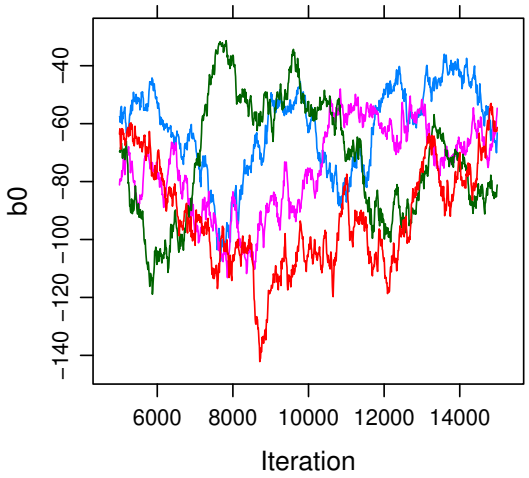
We'll use this to talk about assessing convergence

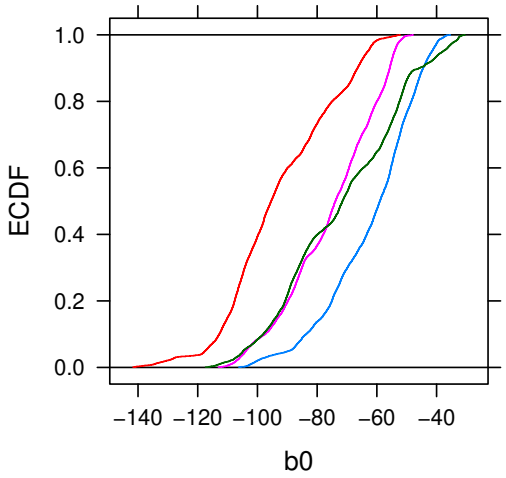
Assessing Convergence using CODA

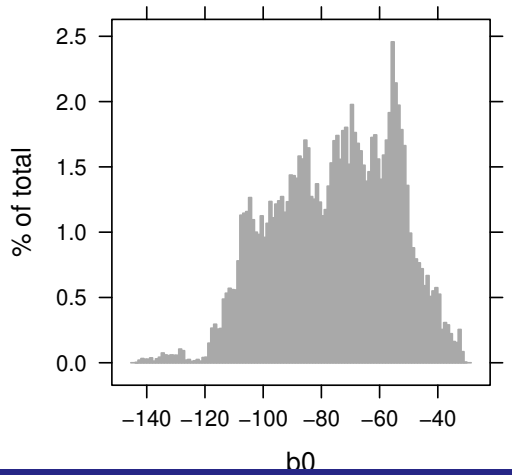
Trace Plots: parameter value $\times$ iteration. These are useful for assessing whether chain have stabilized and see where could set "burn in" or "warm up".

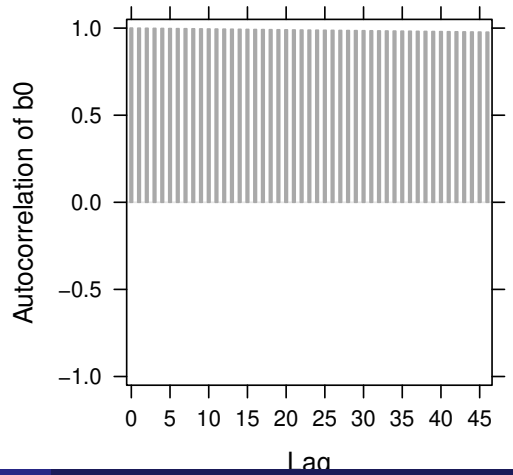
The ones in the anorexia example look good, but the following ones are very bad.

Looks Pretty Bad

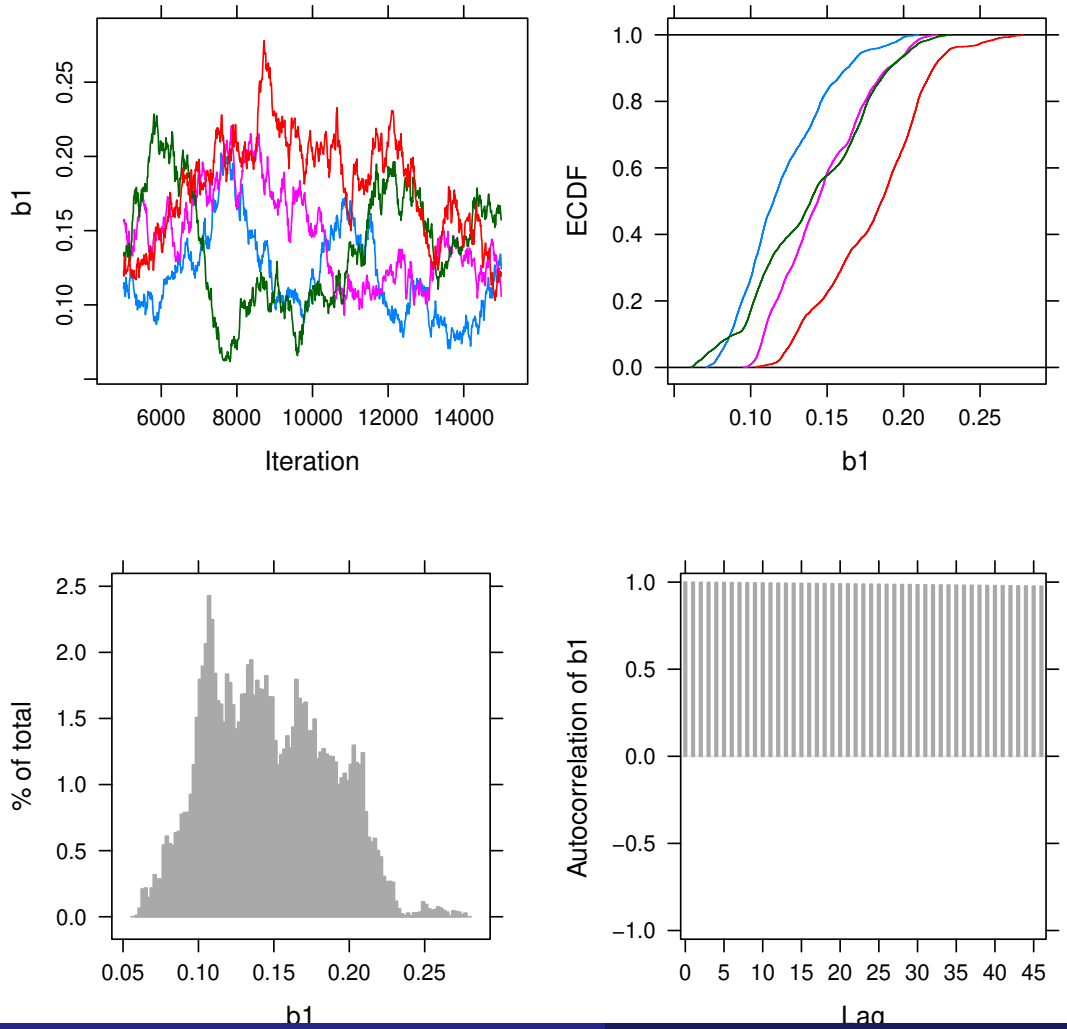








Looks Pretty Bad



Assessing Convergence using CODA

- Geweke Statistic: Test whether the mean of first part of chain (first 10% of θs) equals the mean of the later part of the chain (last 50%). This is based on the assumption that the first and last parts of the chain are (asymptotically) independent, such that difference between the means should be 0. The statistics is $N(0, 1)$.

- Auto-correlations: plot auto-correlations $\times$ iterations. These show the dependencies between candidate θs in a chain. At convergence they should be 0.

Assessing Convergence (continued)

- Effective Sample Size: Even if you have a large number of values in the simulated posterior distribution, due to the dependency between θs we need a correction to the sample size:

$$\text{ESS} = \frac{mn}{1 + 2\sum_t \text{ACF}_t},$$

where ACF_t is the autocorrelation of sequence at lag t, m is 2 times number of chains and n is length of chain.

- Trace plots of multiple chains: Determine whether the chains are mixing well or there is an "orphan". May be the case that it's a bad model or very sensitive to starting values.

Assessing Convergence (continued)

- Gelman-Rubin diagnostic or the "potential scale reductions" or the "shrink factor". The between chain variance relative to the within chain variance should be about the same if all chains have settled. A value > 1.1 is "cause for concern".
  - After deleting warm-ups, split each of the chains into 2, let $n =$ length of split chain, and $m =$ number of split chains.
  - Compute $\bar{\theta}_{.j} = \frac{1}{n}\sum_{i=1}^{n}\theta_{ij}, \qquad \bar{\theta}_{..} = (1/m)\sum_{j=1}^{m}\bar{\theta}_{.j},$

$$
\begin{aligned}
B &= \frac{n}{m-1}\sum_{j=1}^{m}(\bar{\theta}_{.j} - \bar{\theta}_{..})^2 \\
W &= \frac{1}{m}\sum_{j=1}^{m} s_j^2, \text{ where } s_j^2 = \frac{1}{n-1}(\theta_{ij} - \bar{\theta}_j)^2
\end{aligned}
$$

  - $\widehat{\text{var}}(\theta|y)$ is the marginal posterior variance of estimand of θ,

$$\text{PSRF} = \text{Rhat} = \hat{R} = \sqrt{\frac{\frac{n-1}{n}W + \frac{1}{n}B}{W}} = \sqrt{\frac{\widehat{\text{var}}(\theta|y)}{W}}$$

Assessing Convergence (continued)

- Plots of Gelman-Rubin statistics $\times$ iterations.
- Density Estimation: Plot of multiple chains as densities, these should basically be the same.
- High density intervals for multiple chains should all be very similar.
- Descriptive statistics from different simulated distributions for different chains should be similar in value. Note that the Standard Error of Mean can be extended to MCMC using

$$\text{MCSE} = \text{sd}\theta\text{s}/\sqrt{\text{ESS}}$$

runjags

Use the dataList and Model already defined using rjags, but now need multiple starting values, one set per chain.

```
library(runjags)
# Need initial values for each of the 4 chains:
inits1 ← list("mu"=mean(ano$change), "sigma"=sd(ano$change),
          .RNG.name="base::Super-Duper", .RNG.seed=1)
inits2 ← list("mu"=rnorm(1,2,4), "sigma"=1 ,
          .RNG.name="base::Wichmann-Hill", .RNG.seed=2)
inits3 ← list("mu"=rnorm(1,4,1), "sigma"=8 ,
          .RNG.name="base::Wichmann-Hill", .RNG.seed=3)
inits4 ← list("mu"=rnorm(1,-4,2),"sigma"=.5,
          .RNG.name="base::Wichmann-Hill", .RNG.seed=4)

initsList← list(inits1,inits2,inits3,inits4)
```

runjags

```
out.runjags ← run.jags(model=Model1,
      monitor=c("mu","sigma","precision","dic"),
      data=dataList, n.chains=4, inits=initsList)

print(out.runjags)
```

runjags – output

JAGS model summary statistics from 40000 samples (chains = 4; adapt+burnin = 5000)

	Lower95	Median	Upper95	Mean	SD	Mode
mu	0.87629	2.7692	4.6342	2.7661	0.95965	–
sigma	6.8328	8.085	9.5142	8.1331	0.6926	–

runjags – output (continued)

MCerr	MC%ofSD	SSeff	AC.10	psrf
0.0047983	0.5	40000	-0.009977	1.0001
0.0045678	0.7	22991	0.00449	1.0001
0.000016421	0.6	24831	0.0052328	1.0001

Model fit assessment
DIC = 506.5595
PED not available from the stored object
Estimated effective number of parameters: pD = 2.05347
Total time taken: 5 seconds (my desktop)

jagsUI Input

This part is the same as needed for run.jags,

```
# Needs initial values for each of the 4 chains:
initsList = list(list("mu"=mean(ano$change), "sigma"=sd(ano$change)),
          list("mu"=rnorm(1,2,4), "sigma"=1 ),
          list("mu"=rnorm(1,4,1), "sigma"=8 ),
          list("mu"=rnorm(1,-4,2),"sigma"=.5 )
)
```

jagsUI Input (continued)

```
out.jagsUI ← jags(model.file="Model1.txt",
          data=dataList,
          inits=initsList,
          parameters.to.save=c("mu","sigma","precision"),
          n.iter=2000,
          n.burnin=500,
          n.chains=4)

print(out.jagsUI)
```

jagsUI Verbose Output

JAGS output for model 'Model1.txt', generated by jagsUI.
Estimates based on 4 chains of 2000 iterations,
adaptation = 100 iterations (sufficient),
burn-in = 500 iterations and thin rate = 1,
yielding 6000 total samples from the joint posterior.
MCMC ran for 0.013 minutes at time 2018-02-19 17:57:16.

	mean	sd	2.5%	50%	97.5%
mu	2.787	0.959	0.888	2.773	4.678
sigma	8.115	0.701	6.873	8.069	9.618
precision	0.016	0.003	0.011	0.015	0.021
deviance	504.541	2.078	502.520	503.921	509.892

jagsUI Verbose Output

	overlap0	f	Rhat	n.eff
mu	FALSE	0.998	1.001	3759
sigma	FALSE	1.000	1.000	6000
precision	FALSE	1.000	1.000	6000
deviance	FALSE	1.000	1.001	2884

Successful convergence based on Rhat values (all < 1.1). Rhat is the potential scale reduction factor (at convergence, Rhat=1).

For each parameter, n.eff is a crude measure of effective sample size.

overlap checks if falls in the parameter's 95% credible interval.

f is the proportion of the posterior with the same sign as the mean; i.e., our confidence that the parameter is positive or negative.

jagsUI Verbose Output

DIC info: (pD = var(deviance)/2)
pD = 2.2 and DIC = 506.698
DIC is an estimate of expected predictive error (lower is better).

Getting what you paid for

- I got you started, but you'll need to write R for . . .
- Use jags to get estimate of mean and variance of state total SAT scores
- Assess model convergence
- Play with changing priors

Next Up

Mini-outline for rest of semester:

- Linear regression, multiple regression, HLM.
- Handling Missing data
- Evaluating the model itself.
- Metropolis and Metropolis-Hastings Algorithm
- Hamiltonian sampling, Stan, and brms.
- Hierarchial linear regression (i.e., HLM)
- Logistic regression
- Student project presentations.