Gibbs Sampling via jags

Edps 590BAY

Carolyn J. Anderson

Department of Educational Psychology

ILLINOIS

Fall 2019

Overview

- Gibbs sampling for univariate Normal
- Gibbs sampling for more complex problems
- jags
- runjags
- Practice

Depending on the book that you select for this course, read either Gelman et al. pp xx 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.

Why Gibbs?

The advantages of the metropolis algorithm:

- Works well on small problems (few numbers of parameters).
- Very general and can be applied to a wide range of problems.
- Can be incorporated within other MCMC samplers.

The disadvantages:

- Must have symmetric jump (transition) distribution.
- Does not handle larger number of parameters.
- Complex models are difficult.
- Needs to be "tuned".
- Inefficient (i.e., reject some sampled values).

Gibbs sampling

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

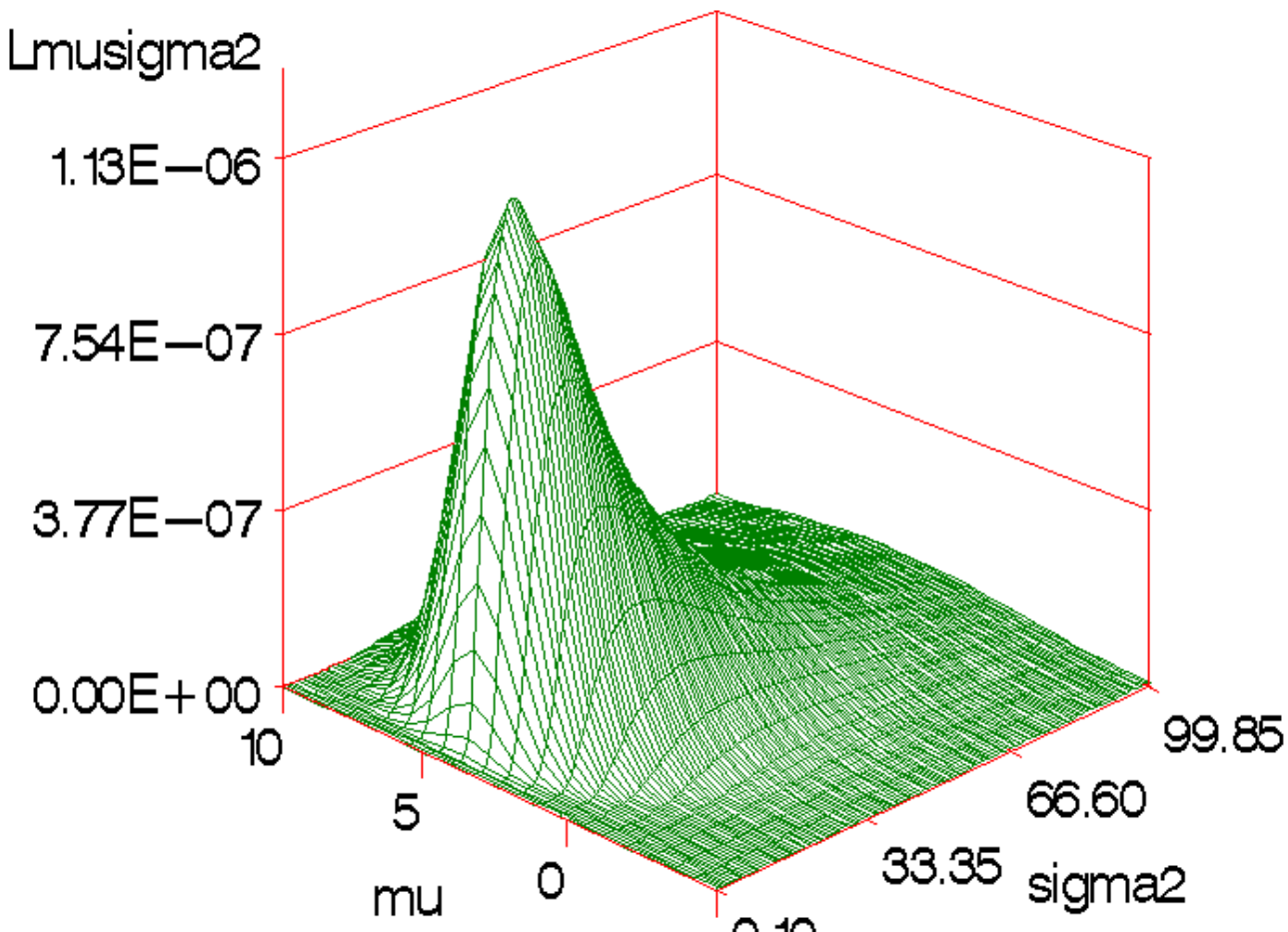
For example, suppose $y \sim N(\mu, \sigma)$ and the posterior is $p(\mu, \sigma|y)$.

1. Sample $\theta^{(t+1)}$ (a value of the mean on the $(t+1)^{\text{th}}$ iteration) from $p(\mu|\sigma^{(t)}, y)$
2. Sample $\sigma^{2(t+1)}$ (a value of the variance on the $(t+1)^{\text{th}}$ iteration) from $p(\sigma^2|\mu^{(t+1)}, y)$

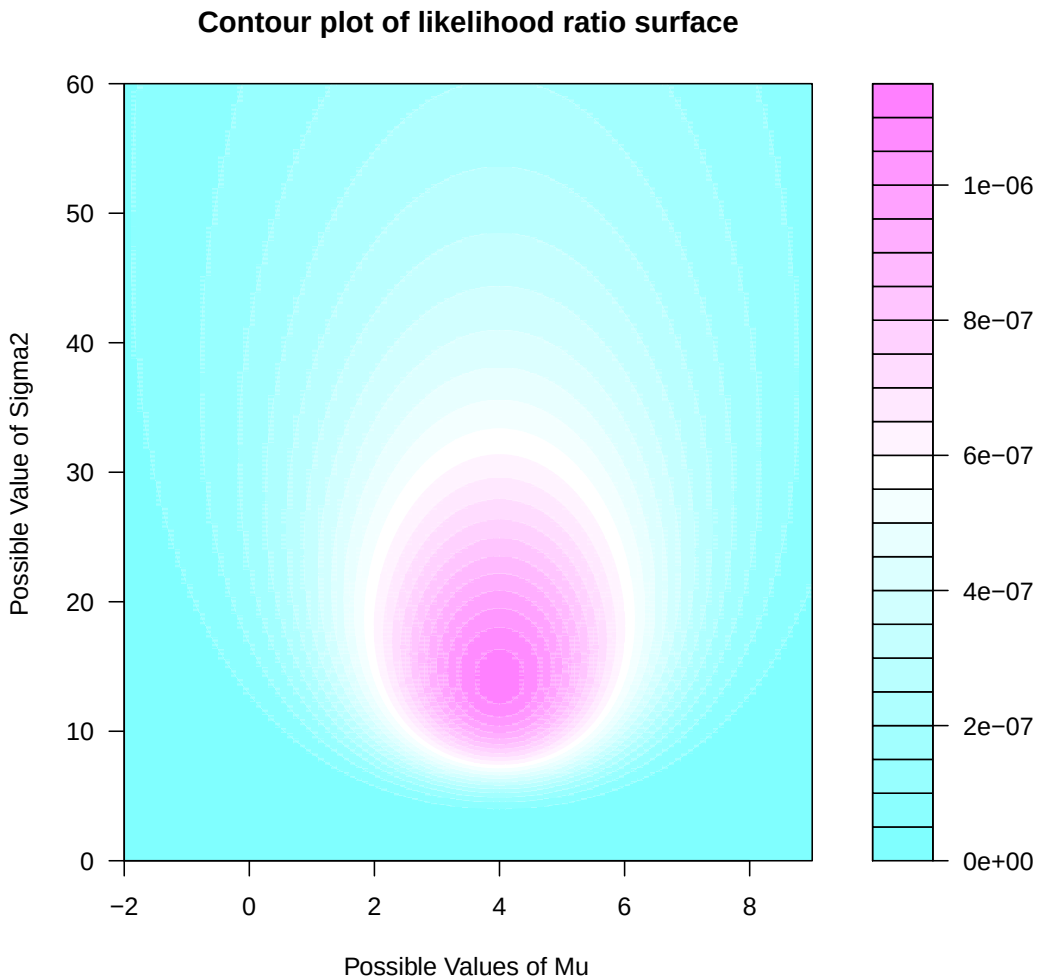
Each time you run steps 1 & 2, you get new values of $\{\theta, \sigma^2\}$ and the repeating these steps many, many times yields an approximation of the posterior distribution, $p(\theta, \sigma|y)$.

A Picture of How It Works: μ and σ^2

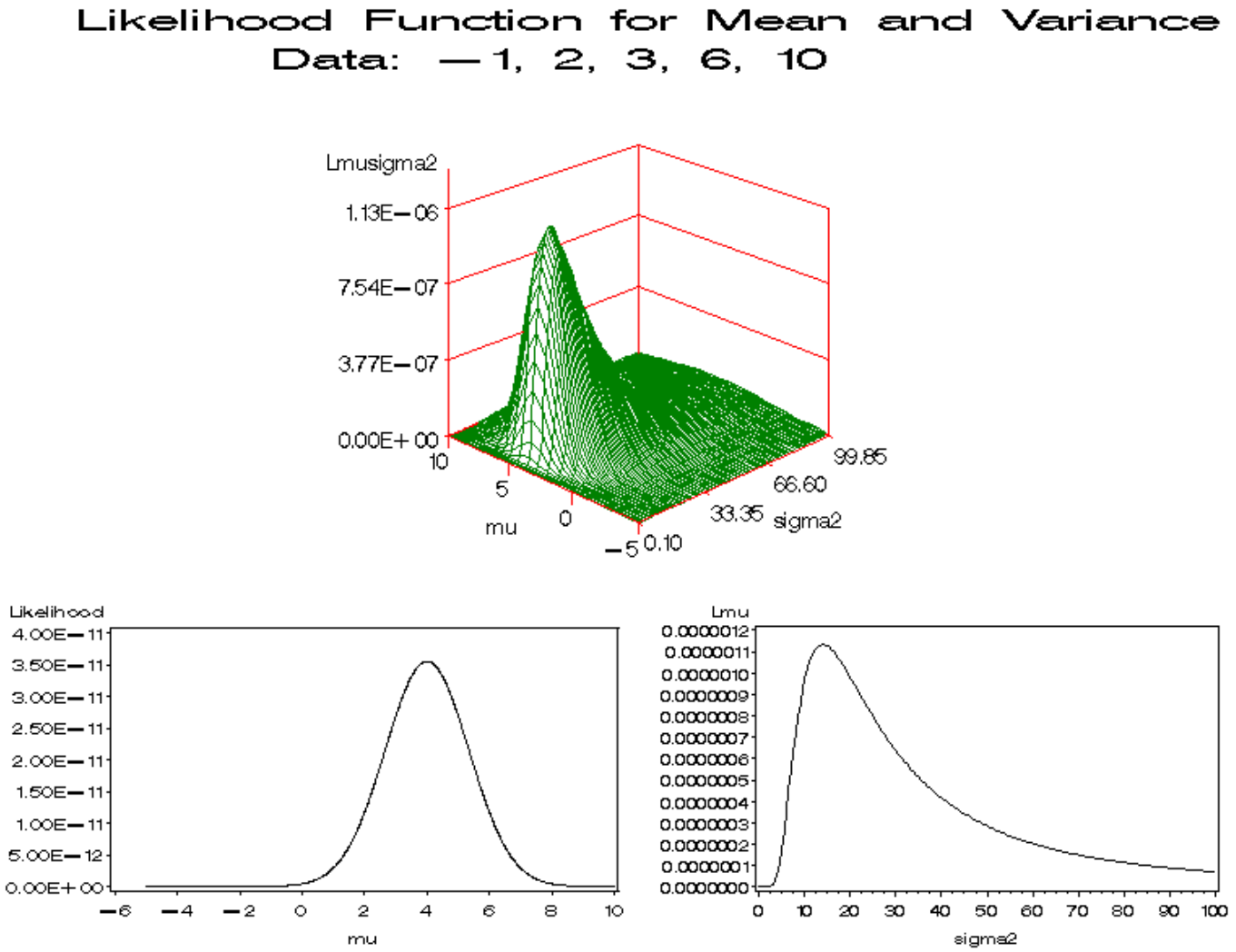
Data: $y_1 = -1, y_2 = 2, y_3 = 3, y_4 = 6, y_5 = 10$



Another View



Univariate Normal: μ and σ^2



$s^2 = \frac{1}{n-1}\sum_i (Y_i - \bar{Y})^2 = 17.5$, $\hat{\sigma}^2 = \frac{1}{n}\sum_i (Y_i - \bar{Y})^2 = 14.0$.

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

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

For the last step, recall that for a normal distribution (in this case a normal prior), μ and σ^2 are independent so things simplify; namely $p(\mu|\tilde{\sigma}^2) = p(\theta|\tilde{\sigma}^2) = p(\mu)$.

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

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

Details for Univariate Normal (continued)

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

where

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

Note:

$$\begin{aligned}
ns_n^2 &= \sum_{i=1}^{n}(y_i - \bar{y} + \bar{y} - \theta)^2 \\
&= \sum_{i=1}^{n}\left((y_i - \bar{y})^2 + 2(y_i - \bar{y})(\bar{y} - \theta) + (\bar{y} - \theta)^2\right) \\
&= (n-1)s^2 + n(\bar{y} - \theta)^2
\end{aligned}$$

And full conditional for θ

$$
\begin{aligned}
\mu|\sigma^2, y_1, \ldots, y_n &\sim N(\mu_n, \tau_n^2) \\
\mu &= \frac{\mu_0/\tau_0^2 + n\bar{y}/\sigma^2}{1/\tau_0^2 + n/\sigma^2} \\
\tau_n^2 &= \left(\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}\right)^{-1} = \frac{1}{\left(\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}\right)}
\end{aligned}
$$

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

Little R function call "gibbs"

Create some data:

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

We also need sample statistics:

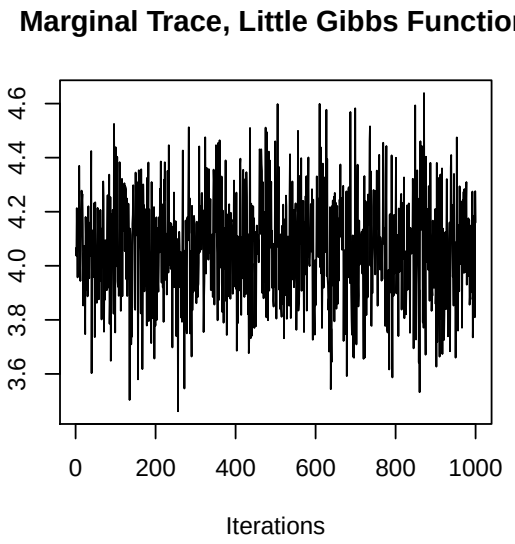
```
ybar ← mean(y)
s2 ← var(y)
```

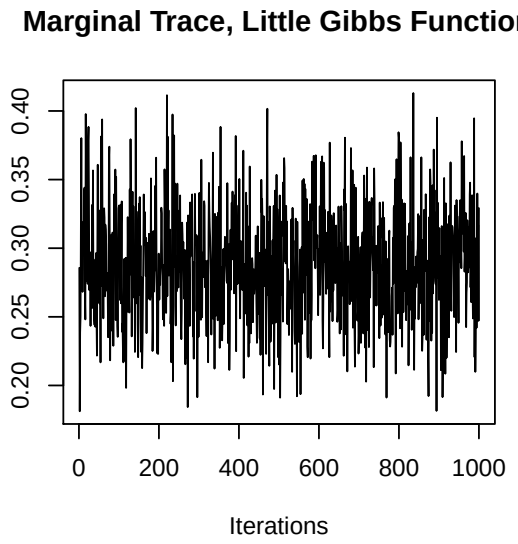
And priors and other things:

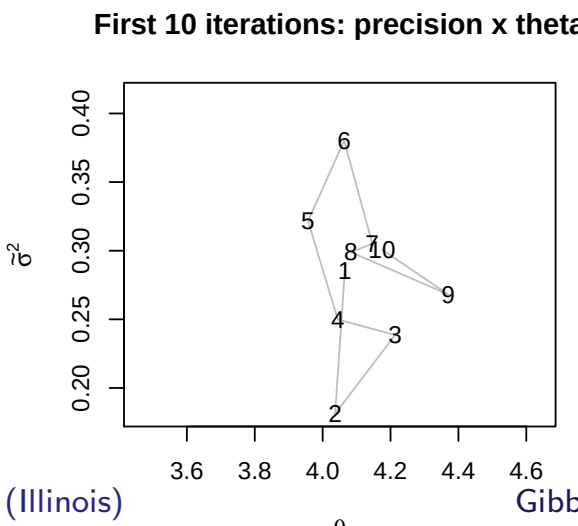
```
mu0 ← 0          t02 ← 100
s02 ← 1          nu0 ← 1
seed ← 2374      S ← 1000
```

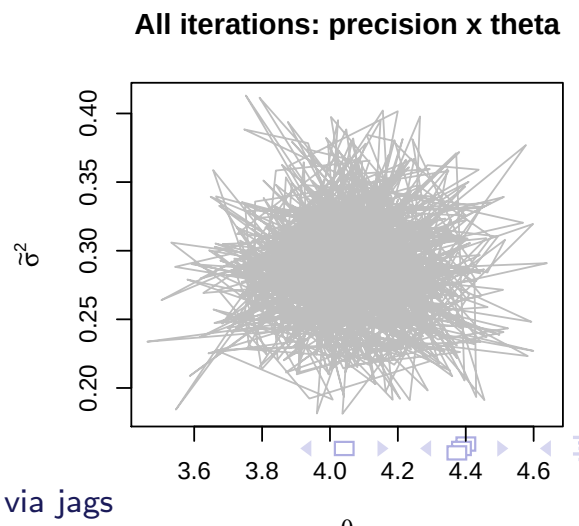
To run it: `sim1 ← gibbs(S,y,mu0,t02,s02,nu0,seed)`

Trace Plots

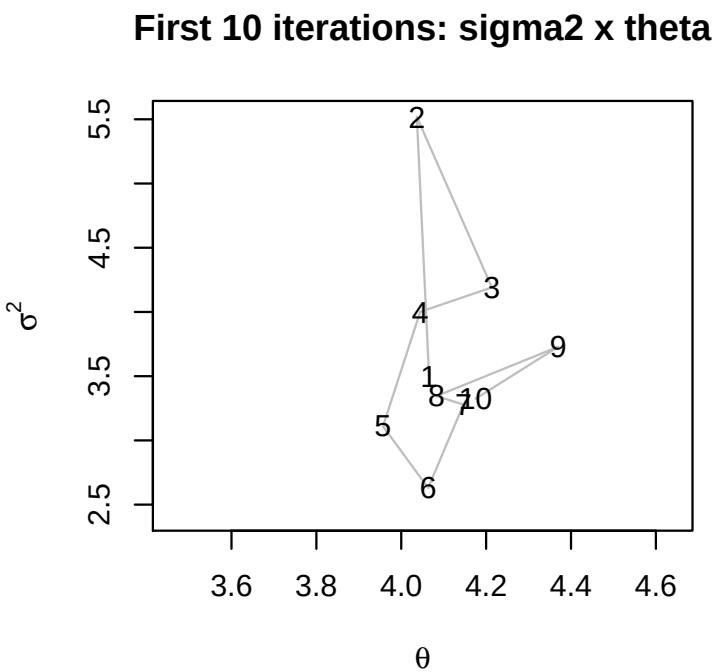

Marginal Trace, Little Gibbs Function
3.6
3.8
4.0
4.2
4.4
4.6
0
200
400
600
800
1000
Iterations

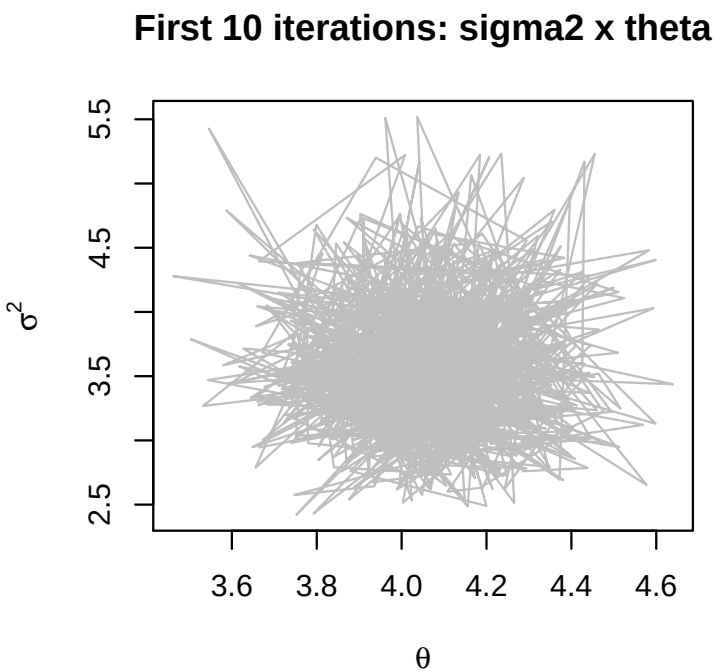

Marginal Trace, Little Gibbs Function
0.20
0.25
0.30
0.35
0.40
0
200
400
600
800
1000
Iterations


First 10 iterations: precision x theta
$\tilde{\sigma}^2$
0.20
0.25
0.30
0.35
0.40
3.6
3.8
4.0
4.2
4.4
4.6
1
2
3
4
5
6
7
8
9
10


All iterations: precision x theta
$\tilde{\sigma}^2$
0.20
0.25
0.30
0.35
0.40
3.6
3.8
4.0
4.2
4.4
4.6

Trace Plots with σ^2





Description of Posterior

Statistics are based on last 50% of iterations.

Estimated Parameters of the Posterior Distribution

θ	τ^2	σ^2	$\sqrt{\sigma^2}$
4.0640	0.1964	3.5490	1.8839

95% High Density intervals:

Parameter	Lower	Upper
θ	3.6557	4.4127
$1/\sigma^2$	0.2111	0.3677
σ^2	2.7193	4.736

What else should we have done?

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 contionalsand sample from each one

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

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

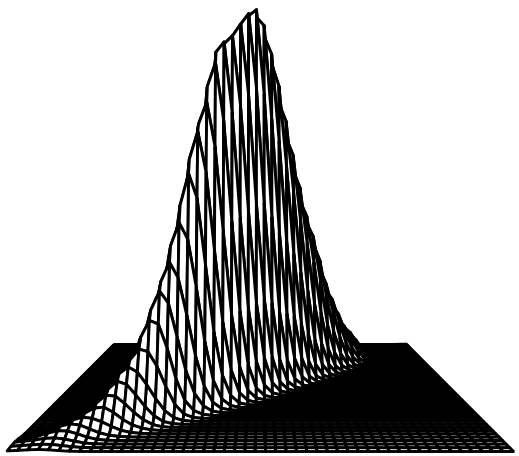
- No tuning needed
- More efficient (don't reject any sample values)
- Can be incorporated into more general MCMC sampler (e.g., Metropolis-Hastings)

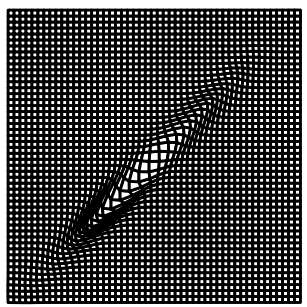
But

- 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)

High Correlation Problem

Below is the density of a bivariate normal with $r \sim .98$.





jags: Implementing Gibbs

We could write functions or R code to implement Gibbs (as done in the 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 ways to implement 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
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)
             )
```

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

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
- dnorm takes as parameters the mean and precision.
- " precision" is precision $= 1/var(y)$.
- "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
```

To run rjags, this is a bit different than my little "gibbs" function.

Compile and Initialize the Model

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

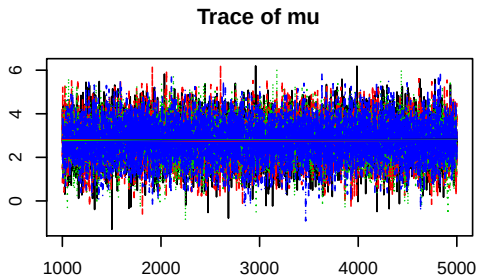
- jags.model compiles and initializes the model.
- I requested 4 chains.
- n.adapt indicates to use adaption during the burn-in or "warm-up" phase; that is, the program self regulates itself to optimize this step.

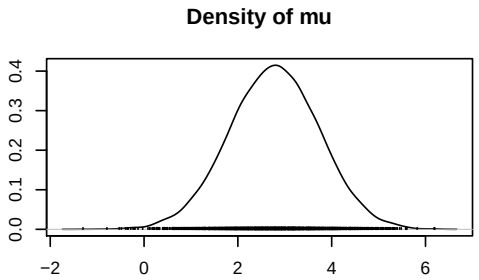
Getting Samples and Summary

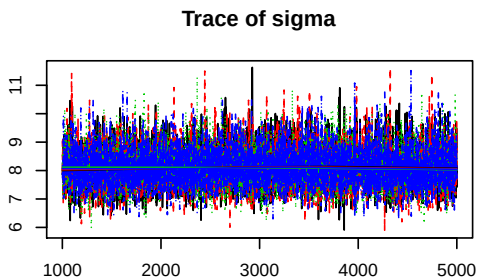
```
# Gets the samples
update (jagsModel1, n.iter=500)

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

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

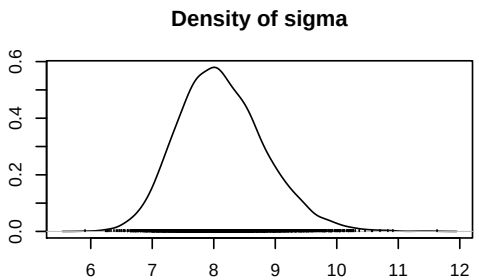
rjags: Trace and Density

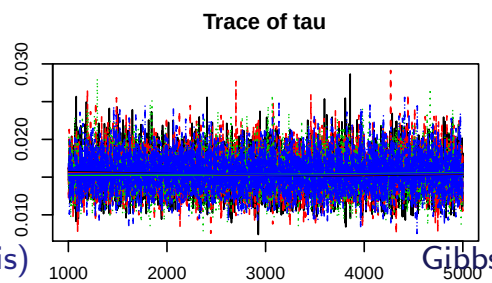
`plot(Samples)`

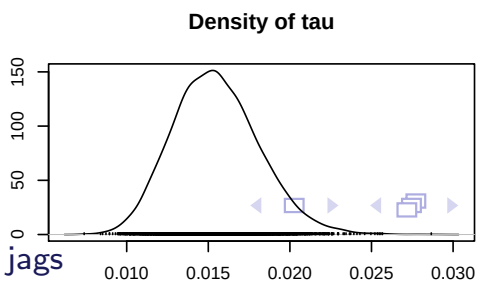




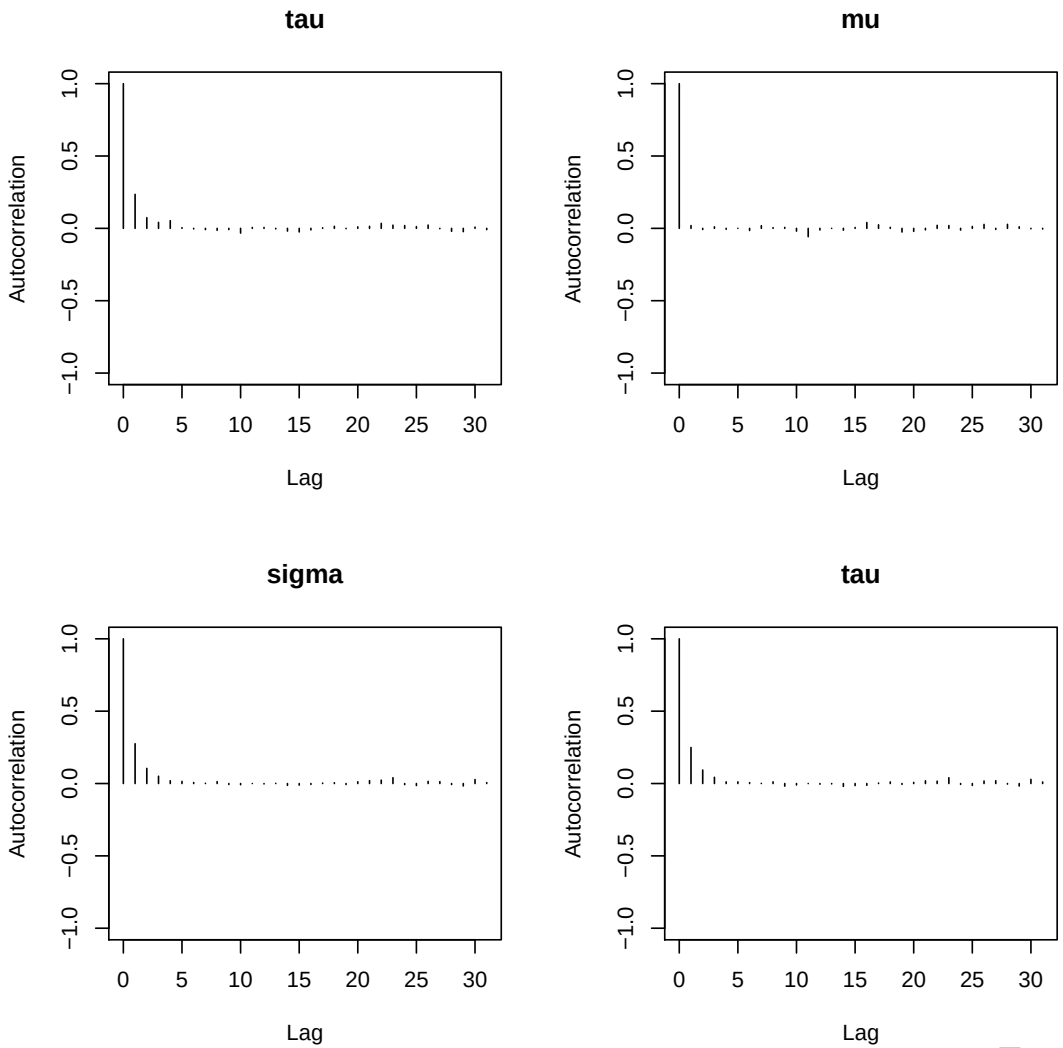




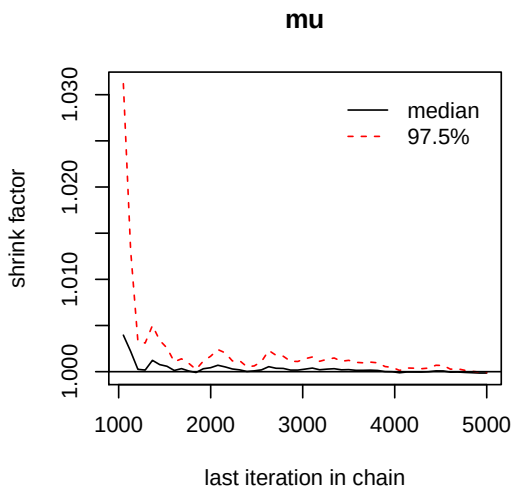


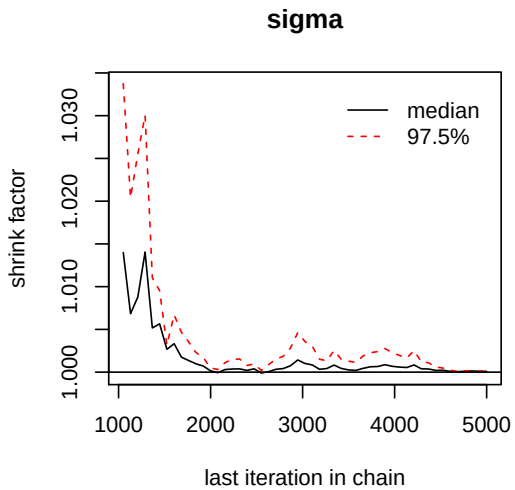


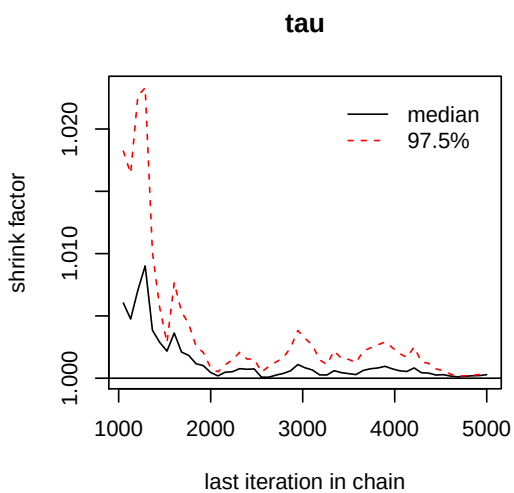
rjags: Auto-Correlation



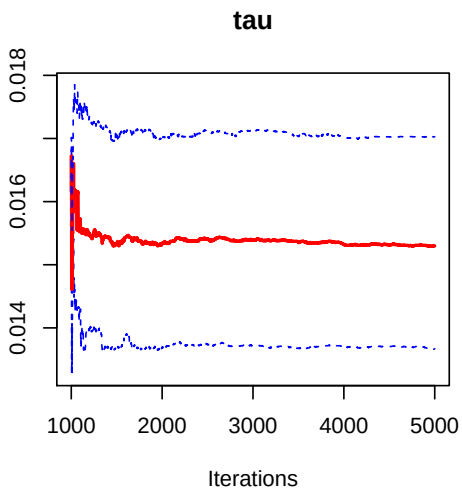
rjags: Gelman

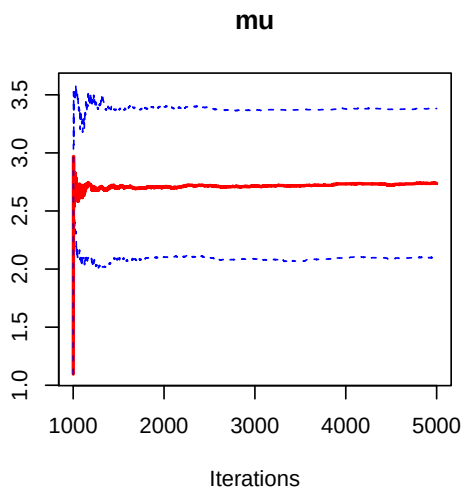


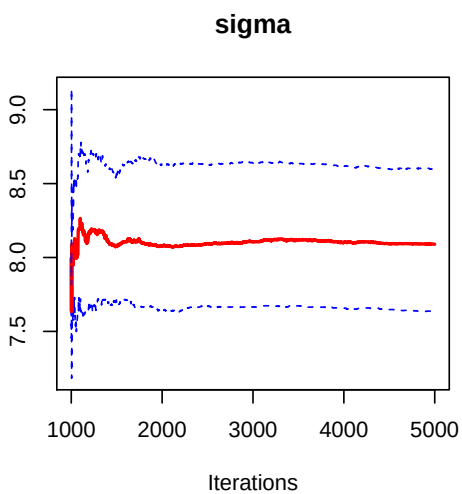


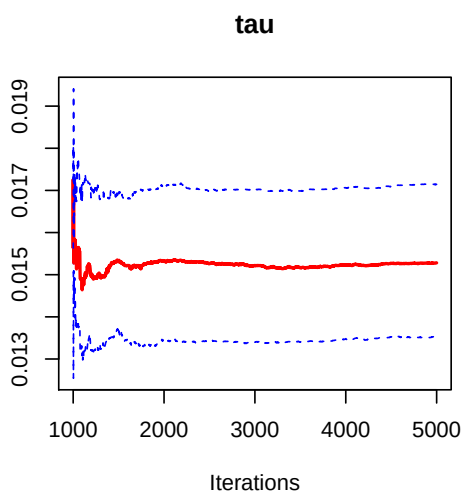


rjags: Cumulative









rjags: Summary Output

```
summary(Samples)
```

Iterations = 1001:5000
Thinning interval = 1
Number of chains = 4
Sample size per chain = 4000

1. Empirical mean and standard deviation for each variable, plus standard error of the mean:

	Mean	SD	Naive SE	Time-series SE
mu	2.77500	0.958299	7.576e-03	7.593e-03
sigma	8.12742	0.705722	5.579e-03	7.382e-03
precison	0.01548	0.002647	2.093e-05	2.695e-05

rjags: Summary Output (continued)

```
summary(Samples)
```

2. Quantiles for each variable:

	2.5%	25%	50%	75%	97.5%
mu	0.88552	2.12940	2.78288	3.42015	4.64054
sigma	6.88867	7.63380	8.07369	8.56890	9.63633
precison	0.01077	0.01362	0.01534	0.01716	0.02107

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.

overlap0 checks if 0 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).

Next Up

Mini-outline for rest of semester:

- Use gibbs to get mean and variance for SAT data.
- Expanding the "model" to linear regression and multiple regression.
- Evaluating the model itself.
- Hamiltonian sampling, Stan, and brms.
- Hierarchial linear regression (i.e., HLM)
- Logistic regression
- Student project presentations.