Posterior Sampling with MCMC

Andy Shen, Devin Francom

LANL: CCS-6

Tasks

Say you have $\mathbf{y}_1, \ldots, \mathbf{y}_n \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where each $\mathbf{y}_i$ is a vector of length p. Use $n = 100$, $p = 3$, $\boldsymbol{\mu} = (1, 2, 3)$ and

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1.0 & 1.4 & 2.1 \\ 1.4 & 4.0 & 4.2 \\ 2.1 & 4.2 & 9.0 \end{pmatrix}$$

to generate some data

Task 1: Use $\mathbf{y}_1, \ldots, \mathbf{y}_n \sim N(\boldsymbol{\mu}, diag(\sigma_1^2, \ldots, \sigma_p^2))$ as your likelihood, with $\boldsymbol{\mu} \sim N(\mathbf{m}, \mathbf{S})$ as your prior for $\boldsymbol{\mu}$ and $\sigma_i^2 \sim InvGamma(a, b)$ as your prior for σ_i^2. Use Gibbs sampling to sample the resulting posterior.

Task 2: Now use $\mathbf{y}_1, \ldots, \mathbf{y}_n \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ as your likelihood, with $\boldsymbol{\mu} \sim N(\mathbf{m}, \mathbf{S})$ as your prior for $\boldsymbol{\mu}$ and the diagonal elements of $\boldsymbol{\Sigma}$ as $\sigma_i^2 \sim InvGamma(a, b)$ and each correlation parameter as $\rho_{ij} \sim Beta(a, b)$. Use Gibbs sampling and Metropolis-Hastings to sample the resulting posterior.

Data Generation

```
set.seed(12)
library(mvtnorm)
library(invgamma)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(truncnorm)
mu <- c(1, 2, 3); mu
```

```
## [1] 1 2 3
```

```
sig <- cbind(c(1, 1.4, 2.1), c(1.4, 4.0, 4.2), c(2.1, 4.2, 9.0)); sig
```

```
##      [,1] [,2] [,3]
## [1,]  1.0  1.4  2.1
## [2,]  1.4  4.0  4.2
## [3,]  2.1  4.2  9.0
```

```
data <- rmvnorm(1000, mu, sig)
n <- nrow(data)
str(data)
```

```
##  num [1:1000, 1:3] -0.0251 -0.6187 0.4604 0.4057 0.3188 ...
```

```
source("mcmc_gibbs_script.R")
```

Task 1

Our likelihood follows a multivariate normal distribution with mean $\boldsymbol{\mu}$ and variance $diag(\sigma_1^2, \ldots, \sigma_p^2)$.

We multiply the likelihood by both priors to get our posterior distribution, $P(\boldsymbol{\mu}, \tilde{\boldsymbol{\Sigma}} \mid \alpha, \beta, \boldsymbol{m}, \boldsymbol{S})$, where $\tilde{\boldsymbol{\Sigma}} = diag(1, 4, 9)$.

We get the following result:

$$P(\boldsymbol{\mu}, \tilde{\boldsymbol{\Sigma}} \mid \alpha, \beta, \boldsymbol{m}, \boldsymbol{S}) \propto exp((\boldsymbol{\mu} - \mathbf{m})' \boldsymbol{S}^{-1} (\boldsymbol{\mu} - \mathbf{m})) \prod_{i=1}^{P=3} \{(\sigma_i^2)^{-\alpha-1} exp(-\frac{\beta}{\sigma_i^2})\} \prod_{i=1}^{100} \{ \prod_{j=1}^{3} [\sigma_j^{2^{-1/2}}] exp\{-\frac{1}{2}(\mathbf{y}_i - \boldsymbol{\mu})' \tilde{\boldsymbol{\Sigma}}^{-1} (\mathbf{y}_i - \boldsymbol{\mu})\}\}$$

Our σ_i^2 values follow an inverse gamma distribution with parameters $\alpha + 50$ and $\beta + \frac{1}{2}\sum_{j=1}^{100}(y_{ji} - \mu_i)^2$

$$\sigma_i^2 \sim IG(\alpha + 50, \beta + \frac{1}{2}\sum_{j=1}^{100}(y_{ji} - \mu_i)^2)$$

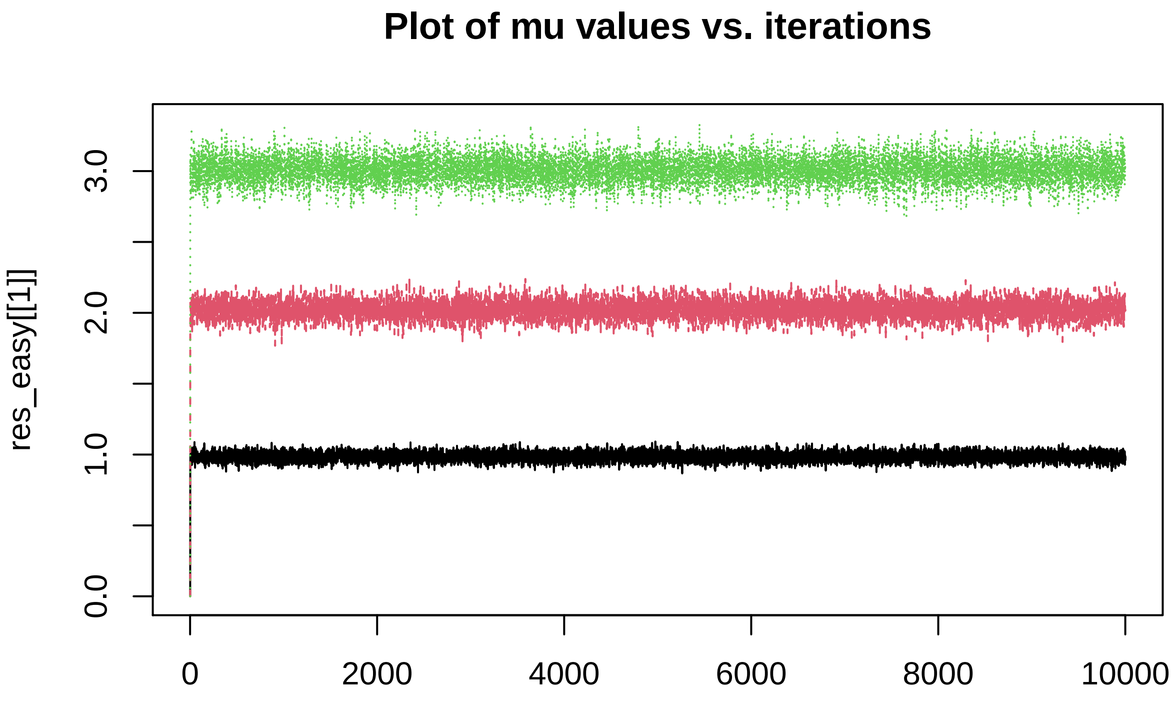
Our μ_i values follow a univariate normal distribution:

$$\mu_i \sim N(\frac{\sigma_i^2}{\sigma_i^2 + 100 s_i^2} m_i + \frac{s_i^2 \sum_{j=1}^{100} y_j}{\sigma_i^2 + 100 s_i^2}, \frac{\sigma_i^2 s_i^2}{\sigma_i^2 + 100 s_i^2})$$

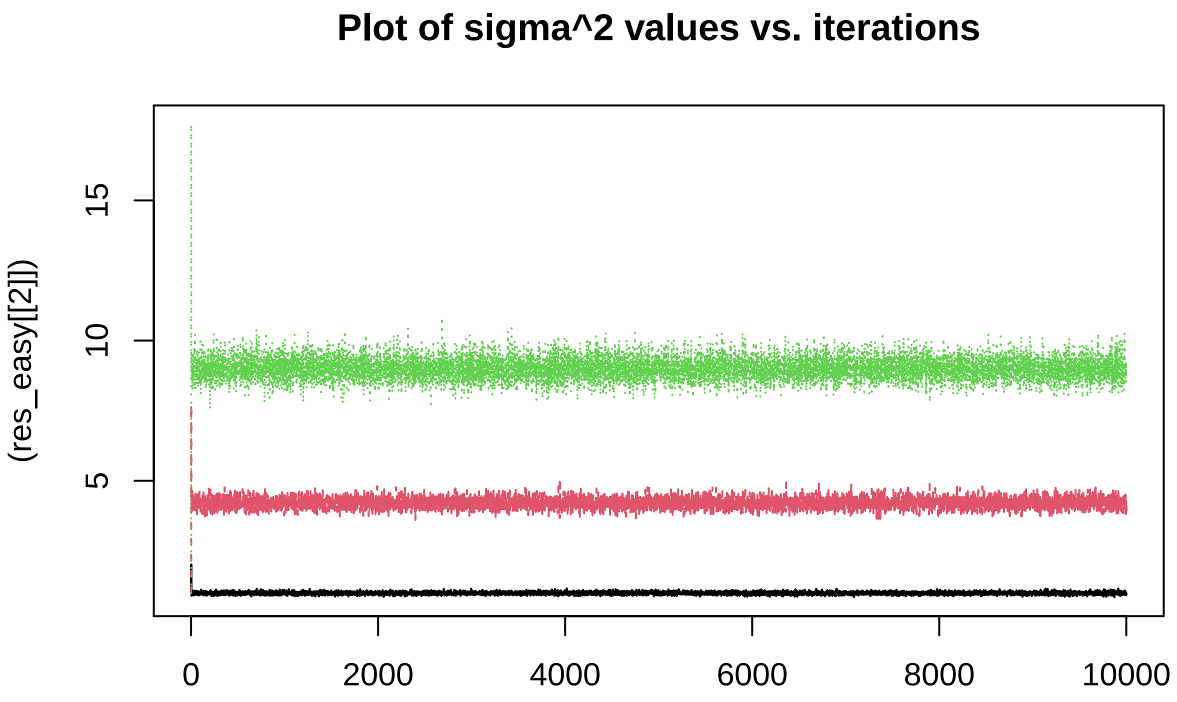
Gibbs Sampling

Please see the `mcmc_gibbs_script.R` file for the code used to generate these results.

```r
res_easy <- gibbs_easy()
matplot(res_easy[[1]], type = "l", main = "Plot of mu values vs. iterations")
```



```r
matplot((res_easy[[2]]), type = "l", main = "Plot of sigma^2 values vs. iterations")
```



Task 2: The Hard Task

Our likelihood follows a multivariate normal distribution with mean $\boldsymbol{\mu}$ and variance $diag(\sigma_1^2, \ldots, \sigma_p^2)$. We multiply the likelihood by both priors to get our posterior distribution, $P(\boldsymbol{\mu}, \boldsymbol{\Sigma} \mid \alpha, \beta, \boldsymbol{m}, \boldsymbol{S})$, where

$$\boldsymbol{\mu} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

and

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1.0 & 1.4 & 2.1 \\ 1.4 & 4.0 & 4.2 \\ 2.1 & 4.2 & 9.0 \end{pmatrix}$$

Our values of $\boldsymbol{S}$ and $\boldsymbol{m}$ used in the prior of $\boldsymbol{\mu}$.

```r
S <- diag(3)
Sinv <- solve(S)
m <- c(0,0,0)
```

It follows that

$$\boldsymbol{\mu} \mid \cdot \quad \sim \quad \mathcal{N}\big[\ (\mathbf{S}^{-1} + n\boldsymbol{\Sigma}^{-1})^{-1}(\mathbf{S}^{-1}\boldsymbol{m} + n\boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{y}}), \quad (\mathbf{S}^{-1} + n\boldsymbol{\Sigma}^{-1})^{-1}\ \big]$$

which is the full conditional distribution for $\boldsymbol{\mu}$.

Recall that for our correlation coefficient ρ_i, we have that

$$\rho_{ij} = \rho_{ji} = \frac{cov(i,j)}{\sigma_i \sigma_j}$$

which we simplify to ρ_i and work with the upper and lower triangular portion of the matrices.

Metropolis-Hastings and Gibbs

Please see the `mcmc_gibbs_script.R` file for the code used to generate these results.

```r
its <- 10000
a <- met_gibbs(its = its)
```

Comparison of Results

Mean Vector

Our acceptance rates of σ_i^2 and ρ_i^2, respectively.

```
a[[4]] / its #sigma
```

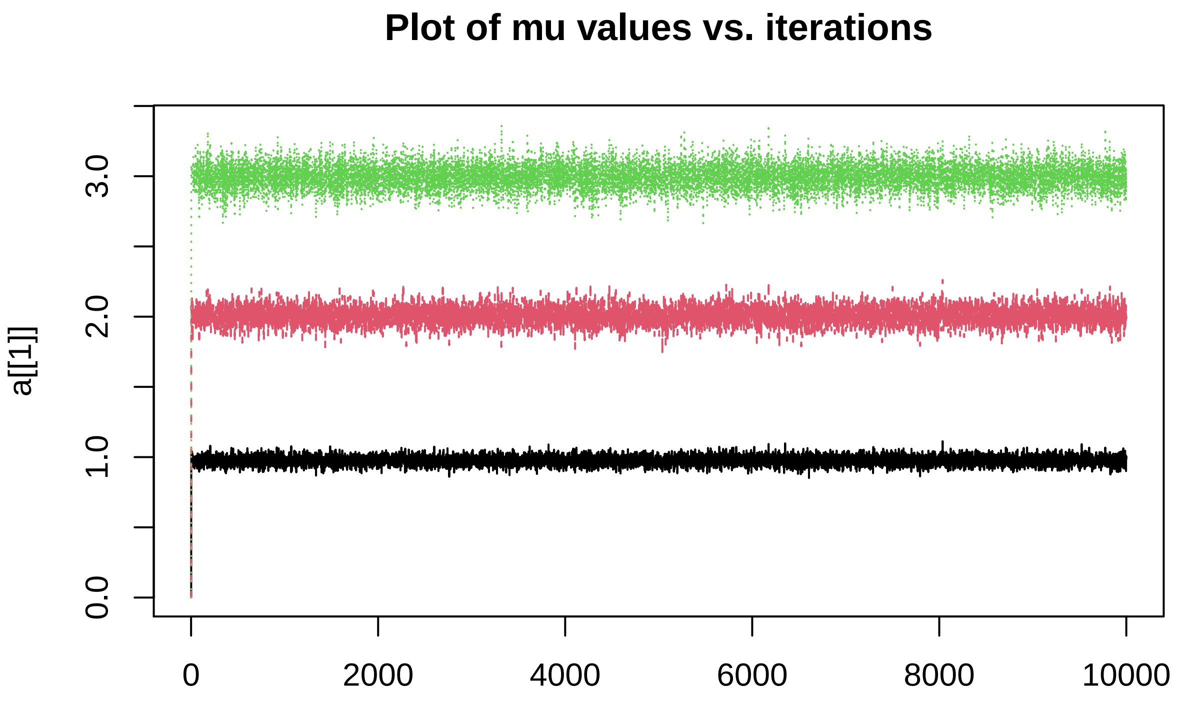
```
## [1] 0.0665 0.2459 0.4555
```

```
a[[5]] / its #rho
```

```
## [1] 0.1648 0.1808 0.1678
```

A plot of our sampled values of $\boldsymbol{\mu}$ is shown below, along with the column means:

```
matplot(a[[1]], type = "l", main = "Plot of mu values vs. iterations")
```



Column means of our sampled values:

```
colMeans(a[[1]])
```

```
## [1] 0.9762275 2.0082311 2.9975778
```

The true mean from the generated data:

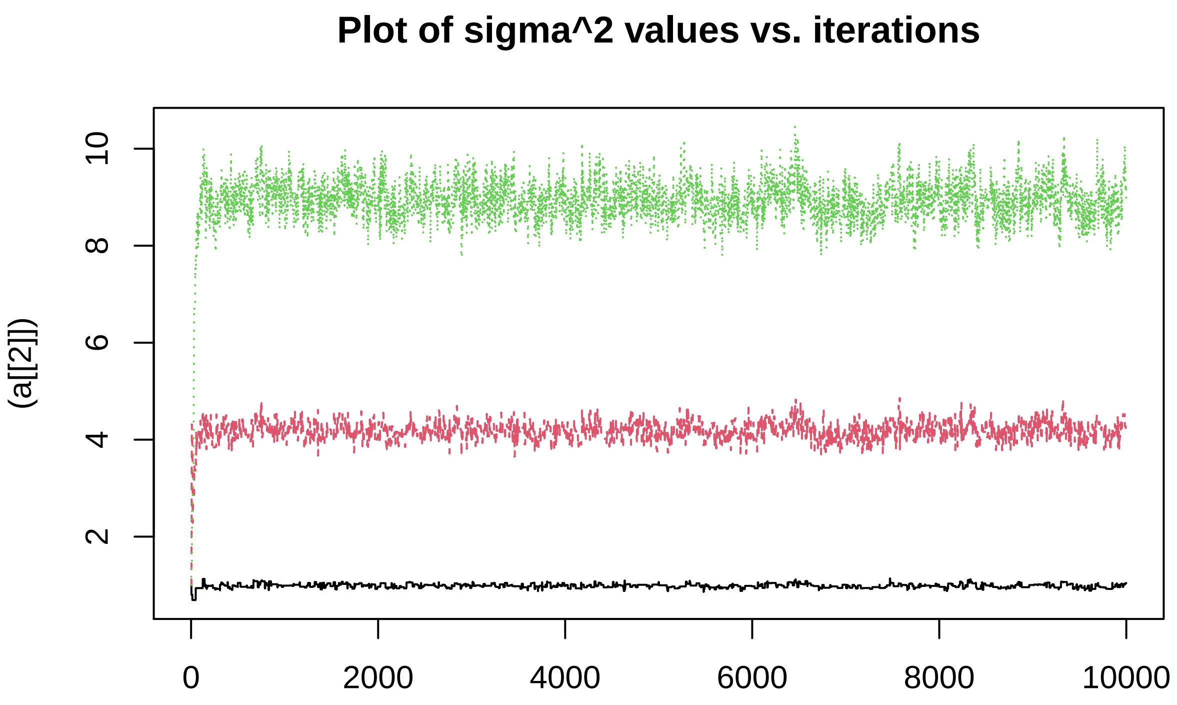
```
colMeans(data)
```

```
## [1] 0.9864302 2.0314982 3.0356240
```

Covariance Matrix

For σ_i^2, the plot of the sampled data and their column means is shown below.

```
matplot((a[[2]]), type = "l", main = "Plot of sigma^2 values vs. iterations")
```



Column means of sampled values:

```
colMeans(a[[2]]) #average sigma^2 values
```

```
## [1] 0.9852891 4.1757176 8.9226997
```

Final covariance matrix $\hat{\Sigma}$

```
.cov <- a[[6]]; .cov #sampled
```

```
##          [,1]     [,2]     [,3]
## [1,] 1.040653 1.465793 2.118412
## [2,] 1.465793 4.076482 4.357852
## [3,] 2.118412 4.357852 8.889966
```

Compare this to the true values:

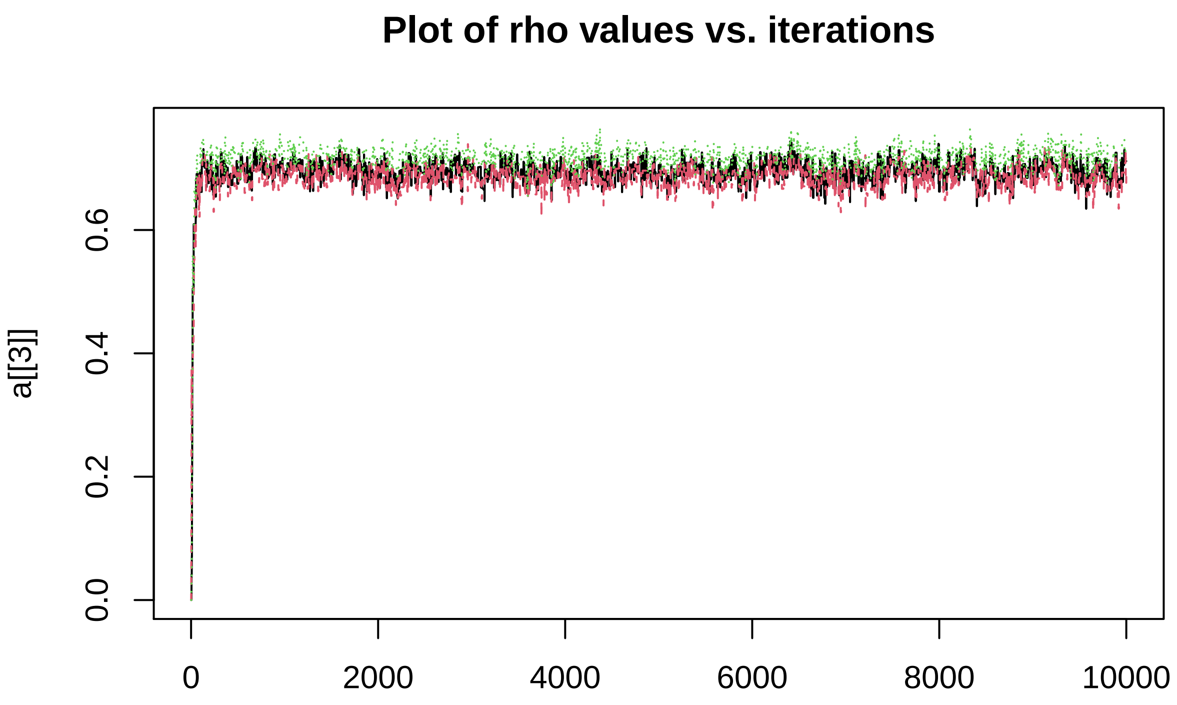
```
cov_data <- cov(data); cov_data #true values
```

```
##           [,1]     [,2]     [,3]
## [1,] 0.9897208 1.429226 2.065696
## [2,] 1.4292258 4.210307 4.454647
## [3,] 2.0656956 4.454647 9.012204
```

Correlation Coefficients

Finally, for ρ_i^2, our results are as follows:

```
matplot(a[[3]], type = "l", main = "Plot of rho values vs. iterations")
```



Column means of sampled values:

```
colMeans(a[[3]]) #average rho values
```

```
## [1] 0.6933825 0.6849516 0.7166481
```

We convert our covariance matrices to correlation matrices of our true and sampled data.

Final values of rho:

```
cor_sampled <- cov2cor(.cov); cor_sampled #sampled values
```

```
##           [,1]      [,2]      [,3]
## [1,] 1.0000000 0.7116671 0.6964782
## [2,] 0.7116671 1.0000000 0.7239018
## [3,] 0.6964782 0.7239018 1.0000000
```

```
cor_sampled[upper.tri(cor_sampled)]
```

```
## [1] 0.7116671 0.6964782 0.7239018
```

Compare this to the true correlation matrix:

```
cor_true <- cov2cor(cov_data); cor_true #true values
```

```
##           [,1]      [,2]      [,3]
## [1,] 1.0000000 0.7001444 0.6916629
## [2,] 0.7001444 1.0000000 0.7231708
## [3,] 0.6916629 0.7231708 1.0000000
```

```
cor_true[upper.tri(cor_true)]
```

```
## [1] 0.7001444 0.6916629 0.7231708
```

As seen here, our sampled values of $\boldsymbol{\mu}$, σ_i^2 and ρ_i closely match the true value from the generated data.