Lab 4: Hierarchical models for Kdrama rating

Author: Shashank Sule

Total Grade for Lab 4: /18

Comments (optional)

Template for lab report

Instructions: This is the template you will use to type up your responses to the exercises. To produce a document that you can print out and turn in just click on Knit PDF above. All you need to do to complete the lab is to type up your BRIEF answers and the R code (when necessary) in the spaces provided below.

It is strongly recommended that you knit your document regularly (minimally after answering each exercise) for two reasons.

1. Ensure that there are no errors in your code that would prevent the document from knitting.
2. View the instructions and your answers in a more legible, attractive format.

```
# Any text BOTH preceded by a hashtag AND within the ```{r} ``` code chunk is a comment.
# R indicates a comment by turning the text green in the editor, and brown in the knitted
# document.
# Comments are not treated as a command to be interpreted by the computer.
# They normally (briefly!) describe the purpose of your command or chunk in plain English.
# However, for this class, they will have a different goal, as the text above and below
# each chunk should sufficiently describe the chunk's contents.
# For this class, comments will be used to indicate where your code should go, or to give
# hints for what the code should look like.
```

Overview

We have explored the Kdrama rating dataset with a hierarchical model. The sampling density, the two-stage prior distribution for μ_j's and the prior distribution for σ are presented below.

- The sampling density for group j, and $j = 1, \cdots, J$:

$$Y_{ij} \overset{i.i.d.}{\sim} \text{Normal}(\mu_j, \sigma), \tag{1}$$

  where $i = 1, \cdots, n_j$ and n_j is the number of observations in group j.

- The stage 1 prior distribution for μ_j :

$$\mu_j \sim \text{Normal}(\mu, \tau). \tag{2}$$

- The stage 2 prior distribution for μ_j:

$$\mu, \tau \sim \text{g}(\mu, \tau). \tag{3}$$

Hyperpriors:

$$\mu \mid \mu_0, \gamma_0 \sim \text{Normal}(\mu_0, \gamma_0), \tag{4}$$
$$1/\tau^2 \mid \alpha_\tau, \beta_\tau \sim \text{Gamma}(\alpha_\tau, \beta_\tau). \tag{5}$$

- The prior distribution for σ:

$$1/\sigma^2 \sim \text{Gamma}(\alpha_\sigma, \beta_\sigma). \tag{6}$$

We have set $\mu_0 = 0.1, \gamma_0 = 0.5, \alpha_\tau = \beta_\tau = \alpha_\sigma = \beta_\sigma = 1$, and obtained posterior summaries below:

	Lower95	Median	Upper95	Mean	SD	Mode	MCerr	MC%ofSD	SSeff	AC.10	psrf
mu	-0.4905	0.1080	0.668	0.1047	0.2884	NA	0.004181	1.4	4758	-0.01052	NA
tau	0.3527	0.6585	1.250	0.7206	0.2749	NA	0.004198	1.5	4288	-0.01511	NA
mu_j[1]	-0.0720	0.0713	0.217	0.0727	0.0724	NA	0.001024	1.4	5000	0.00223	NA
mu_j[2]	-0.0569	0.0994	0.253	0.0993	0.0800	NA	0.001131	1.4	5000	0.01881	NA
mu_j[3]	-0.2415	0.0448	0.349	0.0427	0.1511	NA	0.002073	1.4	5310	-0.00797	NA
mu_j[4]	-0.0248	0.1914	0.399	0.1924	0.1075	NA	0.001520	1.4	5000	-0.01727	NA
sigma	0.2011	0.2616	0.333	0.2650	0.0346	NA	0.000554	1.6	3908	0.00988	NA

We have noticed the issues of negative draws of some parameter which should have been strictly non-negative, including `mu`, `mu_j[1]` through `mu_j[4]`, corresponding to μ and μ_1 through μ_4 in the model. Since these 5 parameters indicate the mean of the mean rating, and the means of ratings, they should be non-negative.

This lab is to explore different prior specifications that would prevent this from happening.

Truncated Normal distributions for the hyperprior for μ and priors for μ_j's

We know that μ and μ_j's should be non-negative. If we still want to use a hyperprior/prior distribution related to the Normal distribution, we can consider the truncated normal distribution.

The truncated Normal distribution

From Wikipedia: The truncated normal distribution is the probability distribution derived from that of a normally distributed random variable by bounding the random variable from either below or above (or both).

Suppose $Y \sim \text{Normal}(\mu, \sigma)$ has a Normal distribution and lies within the interval $Y \in (a, b)$, $-\infty \leq a < b \leq \infty$. Then Y conditional on $a < Y < b$ has a truncated Normal distribution, with pdf:

$$f(y \mid \mu, \sigma, a, b) = \frac{\phi(\frac{y-\mu}{\sigma})}{\sigma\left(\Phi(\frac{b-\mu}{\sigma}) - \Phi(\frac{a-\mu}{\sigma})\right)}, \tag{7}$$

where $\phi(.)$ is the pdf of the standard Normal distribution (i.e. Normal$(0, 1)$)and $\Phi(.)$ is the cdf (cumulative distribution function) of the standard Normal distribution.

Specifying a truncated Normal hyperprior/prior in JAGS

In the previous hierarchical model, where regular Normal prior distribution is assigned to μ_j, the syntax is:

```
for (j in 1:J){
mu_j[j] ~ dnorm(mu, invtau2)
}
```

If we want to use a truncated Normal prior distribution with only non-negative values of μ_j's, one can use the following syntax:

```
for (j in 1:J){
mu_j[j] ~ dnorm(mu, invtau2)T(0,)
}
```

Exercise 1: Give appropriate truncated Normal prior distribution for μ_j's and truncated Normal hyperprior distribution for μ. Run the new hierarchical model, and obtain the posterior summaries for all 7 parameters. Verify that the posterior draws of `mu` and `mu_j[1]` through `mu_j[4]` are all non-negative. Include the 2-by-2 traceplot + cdf + historgram + ACF plot for `mu_j[1]` (Hint: use the `plot(posterior, vars = "mu_j[1]"` command). Comment on the MCMC diagnostics for `mu_j[1]`.

```
knitr::opts_chunk$set(echo = TRUE)
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
require(gridExtra)
```

```
## Loading required package: gridExtra
```

```
require(ProbBayes)
```

```
## Loading required package: ProbBayes
```

```
## Loading required package: LearnBayes
```

```
## Loading required package: dplyr
```

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

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

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

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

```
## Loading required package: shiny
```

```
require(tidyverse)
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.2.1 --
```

```
## v tibble  2.1.3     v purrr   0.3.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::combine() masks gridExtra::combine()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

```r
require(runjags)
```

```
## Loading required package: runjags
```

```
##
## Attaching package: 'runjags'
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```r
require(coda)
```

```
## Loading required package: coda
```

```r
crcblue <- "#2905a1"
knitr::opts_chunk$set(echo = TRUE)
def.chunk.hook  <- knitr::knit_hooks$get("chunk")
knitr::knit_hooks$set(chunk = function(x, options) {
  x <- def.chunk.hook(x, options)
  ifelse(options$size != "normalsize", paste0("\\", options$size,"\n\n", x, "\n\n \\normalsize"), x)
})
```

```r
dramadata = read.csv("KDramaData.csv", header=T)

KBSdrama = dramadata[dramadata$Producer==2,]
KBSdrama$Schedule = as.factor(KBSdrama$Schedule)
```

```r
modelString <-"
model {
## likelihood
for (i in 1:N){
y[i] ~ dnorm(mu_j[schedule[i]], invsigma2)
}

## priors
for (j in 1:J){
mu_j[j] ~ dnorm(mu, invtau2)T(0,)
}
invsigma2 ~ dgamma(a_g, b_g)
sigma <- sqrt(pow(invsigma2, -1))

## hyperpriors
mu ~ dnorm(mu0, 1/g0^2)
invtau2 ~ dgamma(a_t, b_t)
tau <- sqrt(pow(invtau2, -1))
}
"
```

```r
y = KBSdrama$Rating
schedule = KBSdrama$Schedule
N = length(y)
J = length(unique(schedule))

initsfunction <- function(chain){
  .RNG.seed <- c(1,2)[chain]
  .RNG.name <- c("base::Super-Duper",
```

```
                "base::Wichmann-Hill")[chain]
  return(list(.RNG.seed=.RNG.seed,
              .RNG.name=.RNG.name))
}

the_data <- list("y" = y, "schedule" = schedule, "N" = N, "J" = J,
                 "mu0" = 0.1, "g0" = 0.5,
                 "a_t" = 1, "b_t" = 1,
                 "a_g" = 1, "b_g" = 1)
```

```
posterior <- run.jags(modelString,
                      n.chains = 1,
                      data = the_data,
                      monitor = c("mu", "tau", "mu_j", "sigma"),
                      adapt = 1000,
                      burnin = 5000,
                      sample = 5000,
                      thin = 1,
                      inits = initsfunction)
```

```
## Calling the simulation...
## Welcome to JAGS 4.3.0 on Thu Nov  7 19:36:47 2019
## JAGS is free software and comes with ABSOLUTELY NO WARRANTY
## Loading module: basemod: ok
## Loading module: bugs: ok
## . . Reading data file data.txt
## . Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 33
##    Unobserved stochastic nodes: 7
##    Total graph size: 91
## . Reading parameter file inits1.txt
## . Initializing model
## . Adapting 1000
## -------------------------------------------------| 1000
## +++++++++++++++++++++++++++++++++++++++++++++++++ 100%
## Adaptation successful
## . Updating 5000
## -------------------------------------------------| 5000
## ************************************************** 100%
## . . . . . Updating 5000
## -------------------------------------------------| 5000
## ************************************************** 100%
## . . . . Updating 0
## . Deleting model
## .
## Simulation complete.  Reading coda files...
## Coda files loaded successfully
## Calculating summary statistics...
```

```
## Warning: Convergence cannot be assessed with only 1 chain
```

```
## Finished running the simulation
```

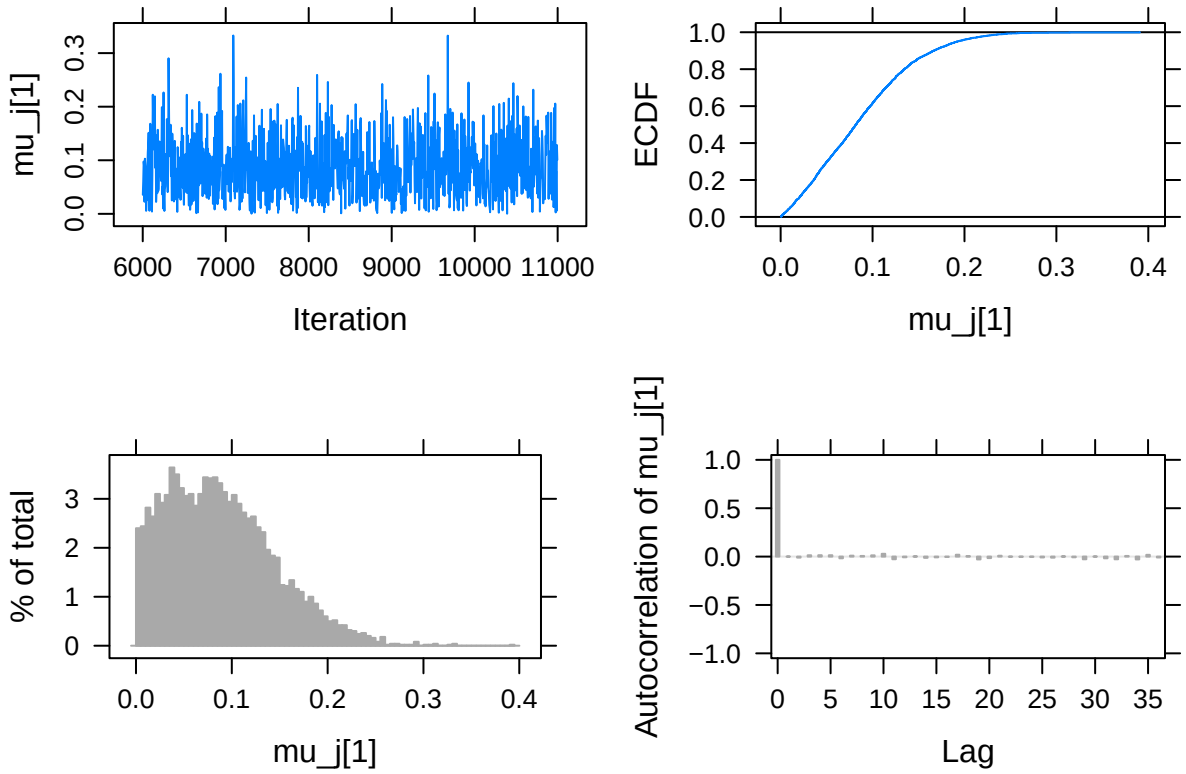
```
summary(posterior)
```

```
##              Lower95      Median  Upper95        Mean         SD Mode
## mu      -1.24994e+00 -0.47595350 0.322263 -0.48800229 0.40383696   NA
## tau      3.53086e-01  0.61910350 1.085760  0.66604317 0.21844532   NA
## mu_j[1]  7.74513e-06  0.08181915 0.191975  0.08823514 0.05668049   NA
## mu_j[2]  5.76010e-04  0.10184400 0.226550  0.10762826 0.06483299   NA
## mu_j[3]  1.55666e-06  0.09836095 0.295068  0.11815688 0.09115240   NA
## mu_j[4]  3.02473e-04  0.18104250 0.354873  0.18727217 0.09714667   NA
## sigma    2.03602e-01  0.25862750 0.327121  0.26183656 0.03301181   NA
##                MCerr MC%ofSD SSeff         AC.10 psrf
## mu      0.0078865099     2.0  2622  0.0030061913   NA
## tau     0.0038093565     1.7  3288 -0.0253720042   NA
## mu_j[1] 0.0008015832     1.4  5000  0.0288199536   NA
## mu_j[2] 0.0009168769     1.4  5000  0.0003873993   NA
## mu_j[3] 0.0013307556     1.5  4692 -0.0111842641   NA
## mu_j[4] 0.0014111140     1.5  4739 -0.0136131723   NA
## sigma   0.0005186738     1.6  4051  0.0076595806   NA
```

Indeed, all the draws of μ_j are strictly non-negative (yet very close to zero).

```
plot(posterior, vars = "mu_j[1]")
```

```
## Generating plots...
```



From the autocorrelation plot, it is clear that there was little correlation between the draws of μ_1. However, it may seem from the CDF and trace plots that the draws weren't exploring the full parameter space. This is because they were being drawn from $N(\mu, \sigma)$ where μ was non-negative but that part of the distribution never got sampled due to truncation.

Grade for Exercise 1: /6

Comments:

Log-normal distributions for the hyperprior for μ and priors for μ_j's

In addition to truncated Normal distribution, we can also consider the log-normal distribution.

The log-normal distribution

From Wikipedia: a log-normal (or lognormal) distribution is a continuous probability distribution of a random variable whose logarithm is Normally distributed. Thus, if the random variable Y is log-normally distributed, then $Y' = \ln(Y)$ has a Normal distribution.

A random variable which is log-normally distributed takes only positive real values, an appealing feature for μ and μ_j's in the Kdrama rating application.

If Y $\sim$ Normal(μ, σ), its pdf is:

$$f(y \mid \mu, \sigma) = \frac{1}{y} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\ln(y) - \mu)^2}{2\sigma^2}\right). \tag{8}$$

Specifying a log-normal hyperprior/prior in JAGS

In the previous hierarchical model, where regular Normal prior distribution is assigned to μ_j, the syntax is:

```
for (j in 1:J){
mu_j[j] ~ dnorm(mu, invtau2)
}
```

If we want to use a truncated Normal prior distribution with only non-negative values of μ_j's, one can use the following syntax:

```
for (j in 1:J){
mu_j[j] ~ dlnorm(mu, invtau2)
}
```

Exercise 2: Give appropriate log-normal prior distribution for μ_j's and log-normal hyperprior distribution for μ. Run the new hierarchical model, and obtain the posterior summaries for all 7 parameters. Verify that the posterior draws of `mu` and `mu_j[1]` through `mu_j[4]` are all non-negative. Include the 2-by-2 traceplot + cdf + historgram + ACF plot for `mu_j[1]` (Hint: use the `plot(posterior, vars = "mu_j[1]"` command). Comment on the MCMC diagnostics for `mu_j[1]`.

```
modelString <-"
model {
## likelihood
for (i in 1:N){
y[i] ~ dnorm(mu_j[schedule[i]], invsigma2)
}

## priors
for (j in 1:J){
```

```
mu_j[j] ~ dlnorm(mu, invtau2)
}
invsigma2 ~ dgamma(a_g, b_g)
sigma <- sqrt(pow(invsigma2, -1))

## hyperpriors
mu ~ dnorm(mu0, 1/g0^2)
invtau2 ~ dgamma(a_t, b_t)
tau <- sqrt(pow(invtau2, -1))
}
"
y = KBSdrama$Rating
schedule = KBSdrama$Schedule
N = length(y)
J = length(unique(schedule))

initsfunction <- function(chain){
  .RNG.seed <- c(1,2)[chain]
  .RNG.name <- c("base::Super-Duper",
                 "base::Wichmann-Hill")[chain]
  return(list(.RNG.seed=.RNG.seed,
              .RNG.name=.RNG.name))
}

the_data <- list("y" = y, "schedule" = schedule, "N" = N, "J" = J,
                 "mu0" = 0.1, "g0" = 0.5,
                 "a_t" = 1, "b_t" = 1,
                 "a_g" = 1, "b_g" = 1)

posterior <- run.jags(modelString,
                      n.chains = 1,
                      data = the_data,
                      monitor = c("mu", "tau", "mu_j", "sigma"),
                      adapt = 1000,
                      burnin = 5000,
                      sample = 5000,
                      thin = 1,
                      inits = initsfunction)
```

```
## Calling the simulation...
## Welcome to JAGS 4.3.0 on Thu Nov  7 19:36:54 2019
## JAGS is free software and comes with ABSOLUTELY NO WARRANTY
## Loading module: basemod: ok
## Loading module: bugs: ok
## . . Reading data file data.txt
## . Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 33
##    Unobserved stochastic nodes: 7
##    Total graph size: 90
## . Reading parameter file inits1.txt
## . Initializing model
```

```
## . Adapting 1000
## -------------------------------------------------| 1000
## ++++++++++++++++++++++++++++++++++++++++++++++++++ 100%
## Adaptation successful
## . Updating 5000
## -------------------------------------------------| 5000
## ************************************************** 100%
## . . . . . Updating 5000
## -------------------------------------------------| 5000
## ************************************************** 100%
## . . . . Updating 0
## . Deleting model
## .
## Simulation complete.  Reading coda files...
## Coda files loaded successfully
## Calculating summary statistics...

## Warning: Convergence cannot be assessed with only 1 chain

## Finished running the simulation
```

```
summary(posterior)
```

```
##              Lower95     Median  Upper95        Mean         SD Mode
## mu      -1.54064e+00 -0.4532665 0.684955 -0.46403423 0.57813396   NA
## tau      4.63178e-01  2.1944750 5.923910  2.61537602 1.68785196   NA
## mu_j[1]  2.90498e-08  0.0649063 0.182129  0.07449279 0.05827772   NA
## mu_j[2]  2.53482e-08  0.0869995 0.209339  0.09274208 0.06531450   NA
## mu_j[3]  2.37677e-07  0.0721666 0.276995  0.09488431 0.08846526   NA
## mu_j[4]  6.20858e-06  0.1547565 0.340284  0.16124374 0.09992770   NA
## sigma    1.99333e-01  0.2577205 0.327432  0.26084767 0.03320195   NA
##                MCerr MC%ofSD SSeff       AC.10 psrf
## mu      0.0189961692     3.3   926 0.081987846   NA
## tau     0.0813073536     4.8   431 0.222720638   NA
## mu_j[1] 0.0019930686     3.4   855 0.097242786   NA
## mu_j[2] 0.0020777432     3.2   988 0.069539785   NA
## mu_j[3] 0.0032845926     3.7   725 0.077482128   NA
## mu_j[4] 0.0024280658     2.4  1694 0.016988154   NA
## sigma   0.0005214649     1.6  4054 0.001690272   NA
```

Grade for Exercise 2: /6

Comments:

Your choice of distribution for the hyperprior for μ and priors for μ_j's

Exercise 3: **Give appropriate prior distribution for μ_j's and hyperprior distribution for μ of your own choosing. Run the new hierarchical model, and obtain the posterior summaries for all 7 parameters. Verify that the posterior draws of `mu` and `mu_j[1]` through `mu_j[4]` are all non-negative. Include the 2-by-2 traceplot + cdf + historgram + ACF plot for `mu_j[1]` (Hint: use the `plot(posterior, vars = "mu_j[1]"` command). Comment on the MCMC diagnostics for `mu_j[1]`.**

Grade for Exercise 3: /6

Comments: