

Bayesian hierarchical modeling

Jingchen (Monika) Hu

MATH 347 Bayesian Statistics

Installing the necessary packages

```
#install.packages("devtools")
require(devtools)
devtools::install_github("bayesball/ProbBayes")

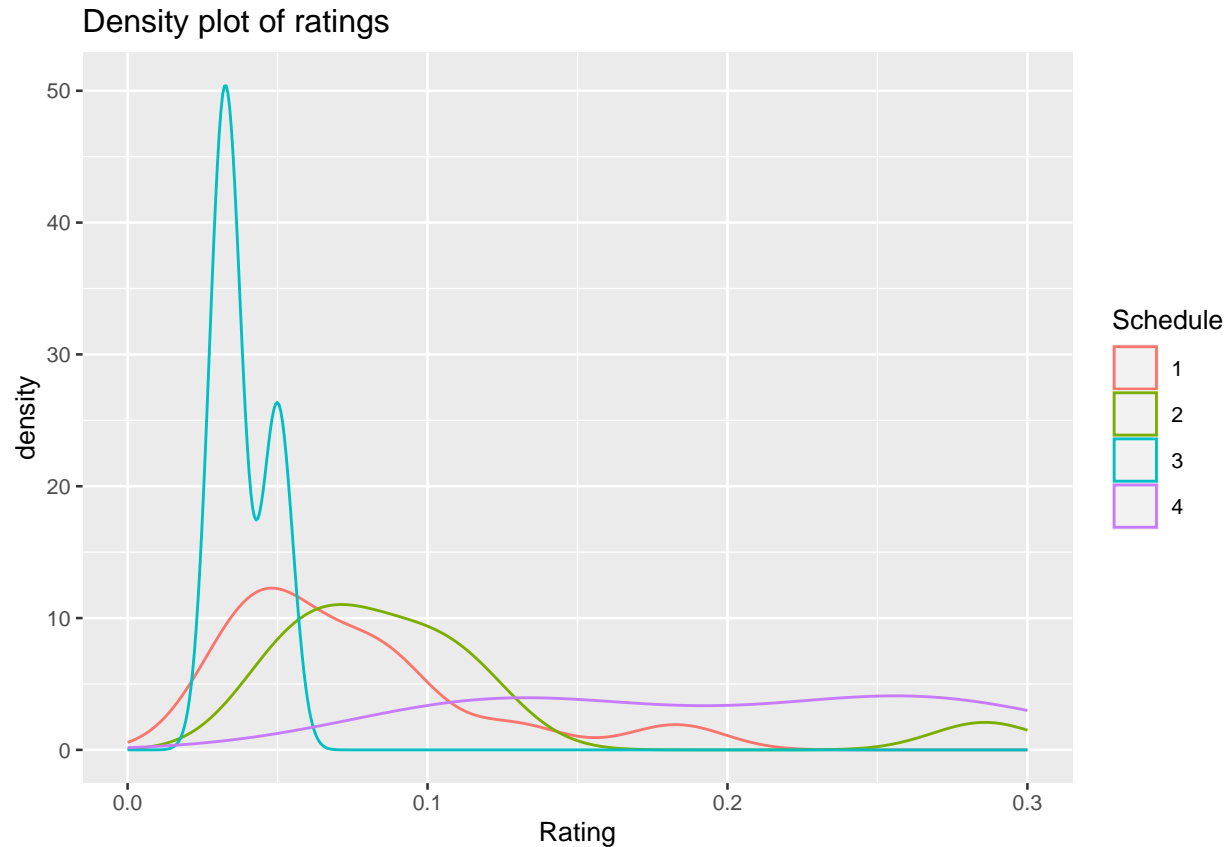
require(ggplot2)
require(gridExtra)
require(ProbBayes)
require(tidyverse)
crcblue <- "#2905a1"
```

Example: Korean Drama Ratings

Ratings by Schedule

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

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



```
table(KBSdrama$Schedule)
```

```
##
##  1  2  3  4
## 13 11  3  6
```

```
tapply(KBSdrama$Rating, KBSdrama$Schedule, summary)
```

```
## $`1`
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.03310 0.04740 0.06310 0.07396 0.08740 0.18290
##
## $`2`
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.04830 0.06465 0.08000 0.09976 0.10645 0.28580
##
## $`3`
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0312  0.0326  0.0340  0.0384  0.0420  0.0500
##
## $`4`
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1120  0.1261  0.1974  0.1949  0.2632  0.2750
```

```
tapply(KBSdrama$Rating, KBSdrama$Schedule, sd)
```

```
##          1          2          3          4
## 0.04264994 0.06608002 0.01014298 0.07570259
```

Observations in groups: approaches to modeling

A two-stage prior in a hierarchical model

The Hierarchical Normal Model

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

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

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

- The stage 1 prior distribution for μ_j :

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

- The stage 2 prior distribution for μ_j :

$$\mu, \tau \sim g(\mu, \tau). \quad (3)$$

- The prior distribution for σ :

$$1/\sigma^2 \sim \text{Gamma}(\alpha_\sigma, \beta_\sigma). \quad (4)$$

Prior and Hyperprior Specifications

- The stage 1 prior distribution for μ_j :

$$\mu_j \sim \text{Normal}(\mu, \tau). \quad (5)$$

- The stage 2 prior distribution for μ_j :

$$\mu, \tau \sim g(\mu, \tau). \quad (6)$$

- Hyperpriors:

$$\mu \mid \mu_0, \gamma_0 \sim \text{Normal}(\mu_0, \gamma_0), \quad (7)$$

$$1/\tau^2 \mid \alpha_\tau, \beta_\tau \sim \text{Gamma}(\alpha_\tau, \beta_\tau). \quad (8)$$

- The prior distribution for σ :

$$1/\sigma^2 \sim \text{Gamma}(\alpha_\sigma, \beta_\sigma). \quad (9)$$

MCMC simulation by JAGS

Recap: Prior and Hyperprior Specifications

- The stage 1 prior distribution for μ_j :

$$\mu_j \sim \text{Normal}(\mu, \tau). \quad (10)$$

- The stage 2 prior distribution for μ_j :

$$\mu, \tau \sim g(\mu, \tau). \quad (11)$$

- Hyperpriors:

$$\mu \mid \mu_0, \gamma_0 \sim \text{Normal}(0.1, 0.5), \quad (12)$$

$$1/\tau^2 \mid \alpha_\tau, \beta_\tau \sim \text{Gamma}(1, 1). \quad (13)$$

- The prior distribution for σ :

$$1/\sigma^2 \sim \text{Gamma}(1, 1). \quad (14)$$

JAGS Script for the Hierarchical Model

```
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)
  }
  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))
}
"
```

Note that `mu_j[schedule[i]]`, we need to determine which j this observation i belongs to. By passing the schedule of each observation as a vector, we can extract which μ_j that we should be working on.

- Notes about the `modelString`

1. Need a vector of μ_j , of length J.
 2. Need a vector of `schedule`, of length N.
 3. `dnorm` takes mean and **precision**.
 4. Work with `invsigma2`, can return `sigma`.
 5. Work with `invtau2`, can return `tau`.
- Pass the data and hyperparameter values to JAGS:

```
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)
```

The `initsfunction` shows the way to set seed with JAGS. `set.seed()` won't work in JAGS.

- Run the JAGS code for this model:

```
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)
```

```
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Note: the model did not require adaptation
## Burning in the model for 5000 iterations...
## Running the model for 5000 iterations...
## Simulation complete
## Calculating summary statistics...
## Finished running the simulation
```

`monitor` is the list of the parameters you want to track.

`adapt` has default 1000, you can think in a way that it's similar to `burnin`.

burnin is the number of draws that you want to toss out.

sample is the final number of iterations that you want to have.

inits is for reproducible setup.

JAGS Output of the Hierarchical Model

- Obtain posterior summaries of all parameters:

```
summary(posterior)
```

##	Lower95	Median	Upper95	Mean	SD	Mode
## mu	-0.49047454	0.10801400	0.6682304	0.10472246	0.28838370	NA
## tau	0.35267052	0.65852607	1.2499367	0.72061727	0.27488885	NA
## mu_j[1]	-0.07196554	0.07132613	0.2167506	0.07266280	0.07238919	NA
## mu_j[2]	-0.05688244	0.09944587	0.2532603	0.09928626	0.07995014	NA
## mu_j[3]	-0.24146558	0.04479396	0.3491075	0.04271370	0.15109887	NA
## mu_j[4]	-0.02481824	0.19140944	0.3991133	0.19242821	0.10745635	NA
## sigma	0.20106992	0.26158432	0.3326434	0.26503794	0.03460242	NA
##	MCerr	MC%ofSD	SSeff	AC.10	psrf	
## mu	0.0041809868	1.4	4758	-0.010518456	NA	
## tau	0.0041977744	1.5	4288	-0.015113465	NA	
## mu_j[1]	0.0010237378	1.4	5000	0.002230488	NA	
## mu_j[2]	0.0011306657	1.4	5000	0.018814005	NA	
## mu_j[3]	0.0020734837	1.4	5310	-0.007974257	NA	
## mu_j[4]	0.0015196623	1.4	5000	-0.017268731	NA	
## sigma	0.0005535073	1.6	3908	0.009879697	NA	

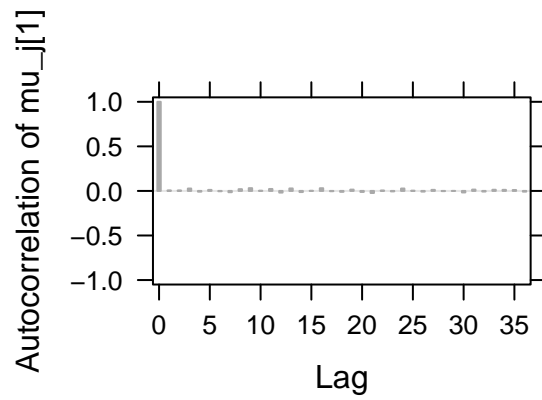
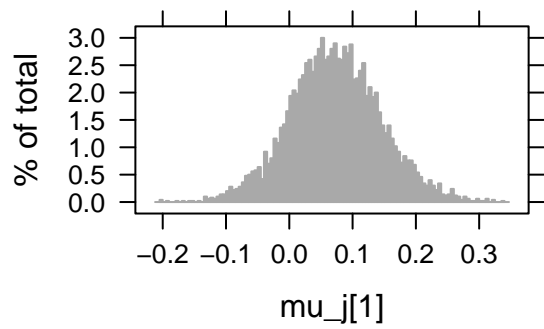
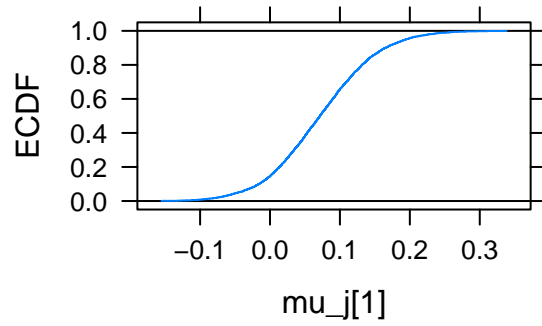
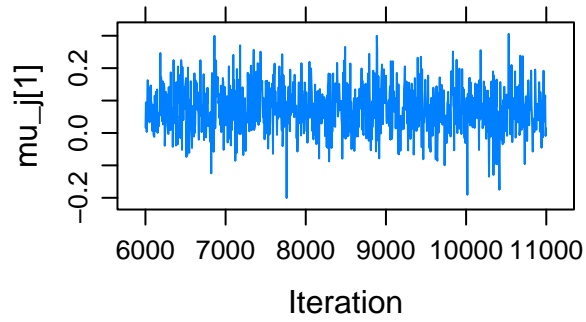
Do you think the μ_j 's (average ratings) do differ a lot across different groups? Look at the mean, they are different.

All credible intervals for μ_j contain 0, which means that they are not statistically significant from 0.

Yet, the lower bounds of credible intervals for μ_j and μ are negative. This doesn't make sense as the rating can not be negative. Using normal may not be suitable for this particular data type. Consider truncated normal.

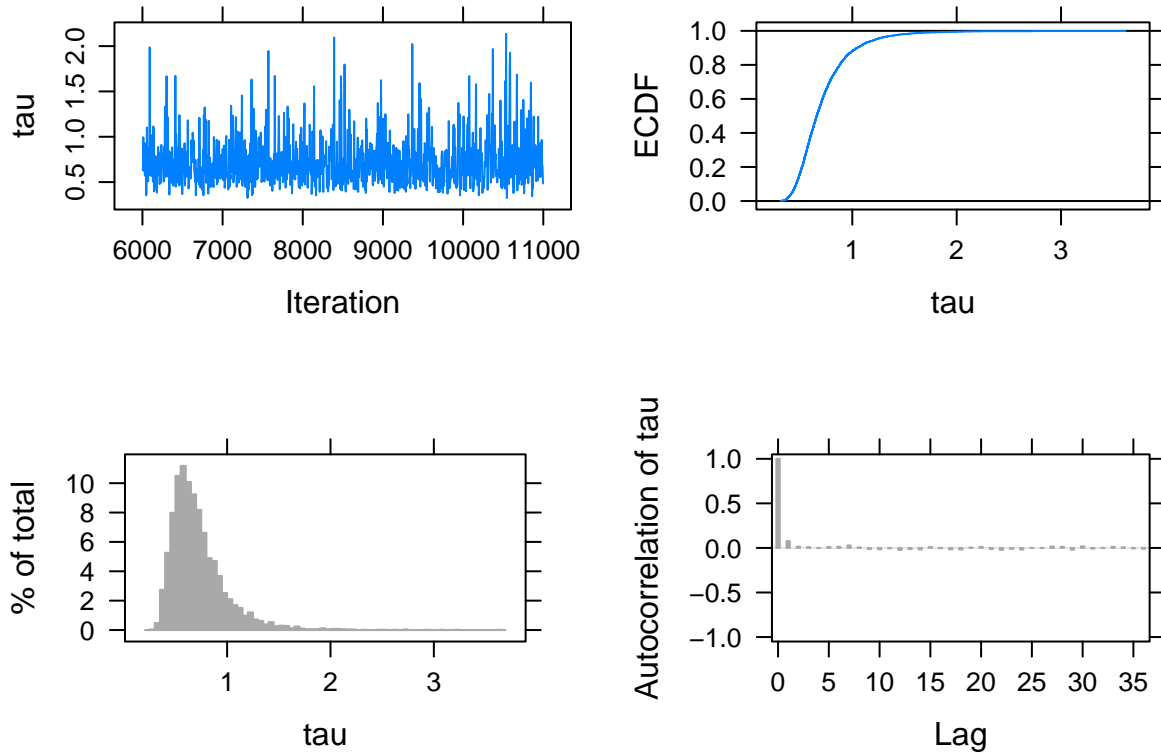
```
plot(posterior, vars = "mu_j[1]")
```

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



```
plot(posterior, vars = "tau")
```

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



Shrinkage/Pooling Effects

```
Ind_Stats = as.data.frame(matrix(NA, J, 2))
names(Ind_Stats) = c("mean", "sd")
for (j in 1:J){
  Ind_Stats[j, ] = c(mean(KBSdrama$Rating[KBSdrama$Schedule == j]),
                     sd(KBSdrama$Rating[KBSdrama$Schedule == j]))
}
```

```
Post_Means <- summary(posterior)[, 4]
```

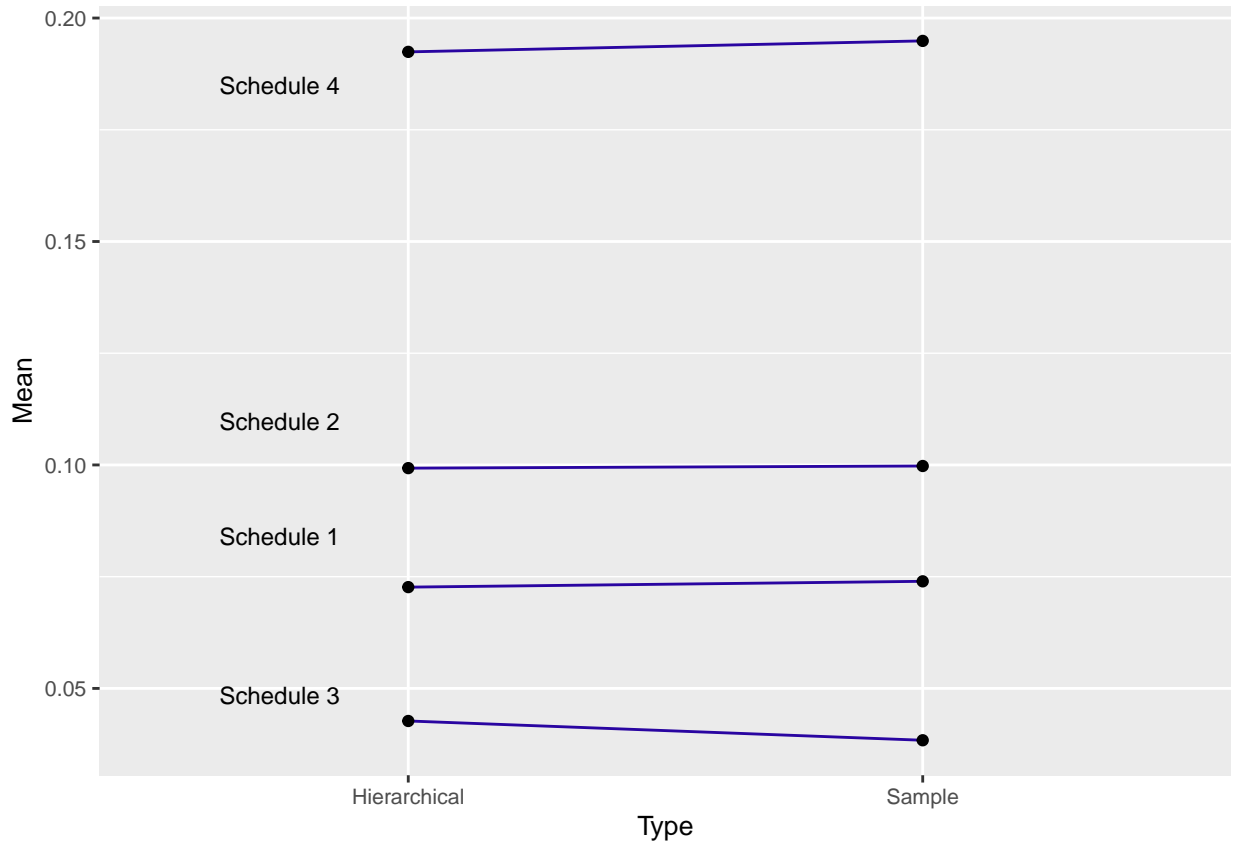
```
Means1 <- data.frame(Type = "Sample", Mean = Ind_Stats$mean)
Means2 <- data.frame(Type = "Hierarchical", Mean =
  Post_Means[3:(4 + J - 2)])
```

```
Means1$Title <- c("Schedule 1", "Schedule 2", "Schedule 3",
  "Schedule 4")
Means2$Title <- c("Schedule 1", "Schedule 2", "Schedule 3",
  "Schedule 4")
```

```
ggplot(rbind(Means1, Means2), aes(Type, Mean, group=Title)) +
  geom_line(color = "blue") + geom_point() +
  annotate(geom = "text", x = 0.75,
```



```
y = Means1$Mean + c(0.01, 0.01, 0.01, -0.01),
size = 3, label = Means1$Title) + increasefont(Size = 10)
```



The right-hand side is the sample mean of the 4 different schedules. On the left-hand side, we have the posterior mean after fitting the hierarchical model.

Sources of Variability

- Two sources of variability in Y_{ij} :

$$Y_{ij} \stackrel{i.i.d.}{\sim} \text{Normal}(\mu_j, \sigma) \text{ [within-group variability]} \quad (15)$$

$$\mu_j \mid \mu, \tau \sim \text{Normal}(\mu, \tau) \text{ [between-group variability]} \quad (16)$$

- To compare these two sources of variability, one can compute the fraction

$$R = \frac{\tau^2}{\tau^2 + \sigma^2}, \quad (17)$$

from the posterior draws of τ and σ .

- The closer the value of R to 1, the higher the between-group variability. ## Compute and Graph Sources of Variability
- We need the `coda` R package

```
#install.packages("coda")
```

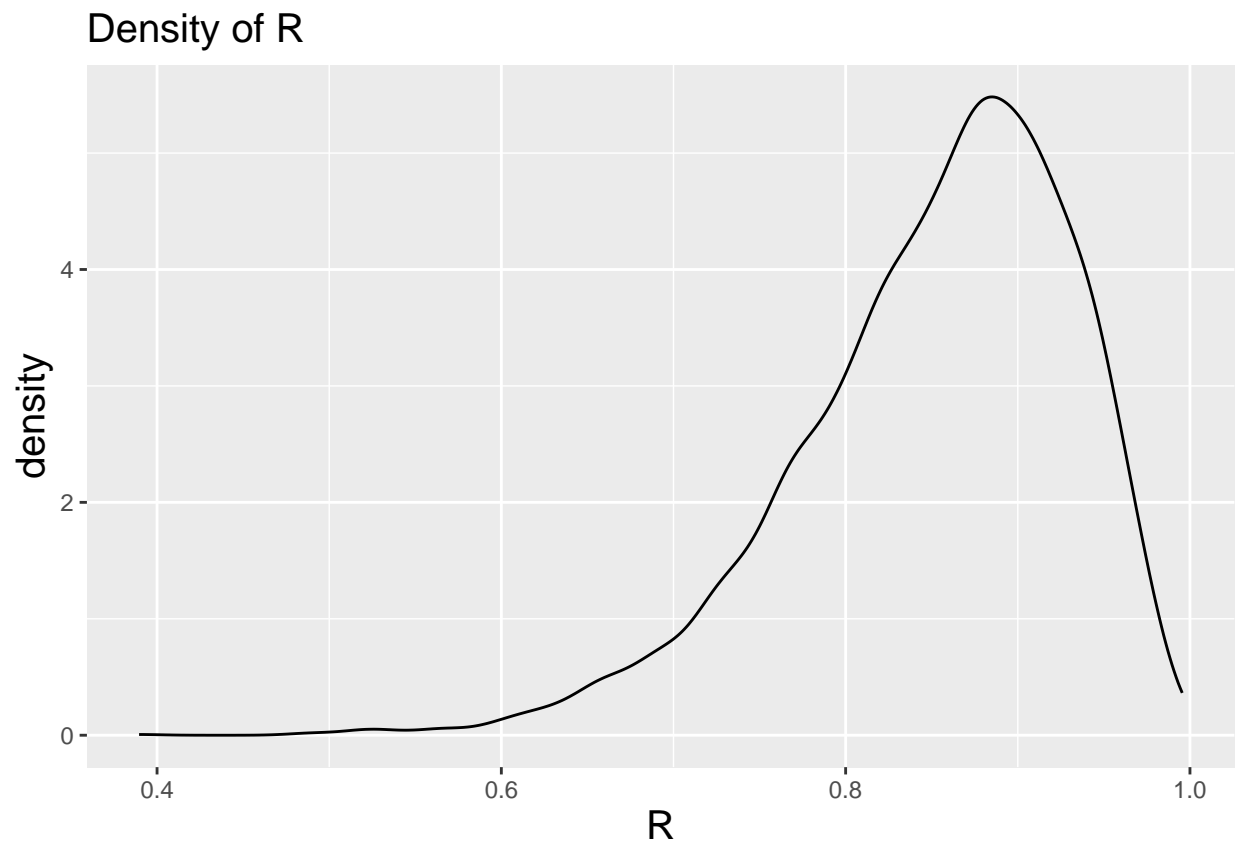
```
require(coda)
tau_draws <- as.mcmc(posterior, vars = "tau")
sigma_draws <- as.mcmc(posterior, vars = "sigma")
R <- tau_draws^2/(tau_draws^2 + sigma_draws^2)

df <- as.data.frame(R)

quantile(R, c(0.025, 0.975))
```

```
##      2.5%      97.5%
## 0.6620553 0.9687803
```

```
ggplot(df, aes(x=R)) + geom_density() +
  labs(title="Density of R") +
  theme(plot.title = element_text(size=15)) +
  theme(axis.title = element_text(size=15))
```



Exercise: Hierarchical model with schedule-specific μ_j and σ_j

P.37/42

- Specify a shared Normal prior distribution for μ_j

$$\mu_j \sim \text{Normal}(\mu, \tau) \quad (18)$$

We think the μ_j 's are related to each other, but they are not the same. We think all of them come from normal distribution with mean μ (they should be somewhere around μ basically). τ controls how μ_j are going to be different from μ or different from each other.

- In addition, specify a shared Gamma prior distribution for σ_j

$$\frac{1}{\sigma_j^2} \sim \text{Gamma}(s_a, s_b) \quad (19)$$

where the mean is s_a/s_b and the variance is s_a/s_b^2

All of the σ_j 's are related somehow. The mean controls on average how they should be. The variance controls the similarity between the σ_j 's.

```
modelString <-"
model {
  ## likelihood
  for (i in 1:N){
    y[i] ~ dnorm(mu_j[schedule[i]], invsigma2_j[schedule[i]])
  }

  ## priors
  for (j in 1:J){
    mu_j[j] ~ dnorm(mu, invtau2)
    invsigma2_j[j] ~ dgamma(s_a, s_b)
  }

  ## sigma <- sqrt(pow(invsigma2, -1))

  ## hyperpriors
  mu ~ dnorm(mu0, 1/g0^2)
  invtau2 ~ dgamma(a_t, b_t)
  tau <- sqrt(pow(invttau2, -1))
}
"
```

Recap