Bayesian Hierarchical Model Analysis of Movie Ratings

Author: Tian Sun, Ted Cigler, Richard Ma

2022-12-11

Contents

Abstract

The MovieLens dataset provides a user-driven alternative to box office statistics for assessing the performance of a movie. In this report, we generate a 2-stage Bayesian hierarchical model Analysis of movie ratings. With JAGS, we grow inference through MCMC and find three posteriors with their distributions that estimate the probability for certain movie score.

Introduction

Filmmaking is a multi-billion dollar industry that relies on only cursory statistics that boil down mainly to the amount of money that each movie earns compared to how much it costs. This presents an opportunity to approach the industry from a different perspective, analyzing a different type of response to the quality of the film to provide a never-before-used model for the success of a film. To this end, we have employed the data from MovieLens, which is a collection of 100 thousand ratings of nine hundred movies from six hundred users, of which we will use the subset of all "Comedy" movies. Making movies is a costly process, and providing in-depth analysis of the performance of movies could help to prevent a "flop" or a significant outlier that performs particularly poorly. The people making a movie want to have some level of assurance that their time and money will be well compensated. Through this analysis, we can find distributions for posteriors to inference the probability that a movie receives a certain score, based on the distribution of the previous ratings of different movies.

Statistical Methods

We utilize a 2-stage hierarchical modele resulting in this posterior model:

- sampling for $j = 1,\ldots,$ 2488 as different movies, $i = 1,\ldots,n_j$ as the number of ratings for movie j:

$$Y_{ij}|\mu_j, \sigma_j \overset{iid}{\sim} N(\mu_j, \sigma_j^2).$$

Note that μ_j and σ_j are hyperparameters. Since we have filtered all the movies to be in the "Comedy" genre and the ratings are in a short range of 1 to 5 we assume a common value of the standard deviation σ across movies. Hence, our new model will be:

$$Y_{ij}|\mu_j, \sigma \overset{iid}{\sim} N(\mu_j, \sigma^2).$$

Next, we set prior distributions for μ_j and σ. For the prior distribution of μ_j, recall again the whole movies share the same genre, it is reasonable to believe that the mean ratings are similar across movies. Hence, we could set up a first-stage prior normal distribution for μ_j.

- first-stage prior for μ_j :

$$\mu_j|\mu, \tau \overset{iid}{\sim} N(\mu, \tau^2).$$

The hyperparameters μ and τ are treated as random since we are unsure about the pooling of ratings. The mean μ and standard deviation τ in this Normal prior distribution reflect respectively the mean and spread of the mean ratings across different movies. We could assign two independent prior distributions: a Normal distribution for μ and an inverse Gamma distribution for τ. These conditional posterior distributions of two parameters helps the Gibbs sampling.

- second-stage prior for μ:

$$\mu|\mu_0, s_0 \sim N(\mu_0, s_0^2).$$

- second-stage prior for τ:

$$1/\tau^2|\alpha_1, \beta_1 \sim Gamma(\alpha_1, \beta_1).$$

Similarly, we could set a prior for σ with gamma on the inverse of variance:

- prior for σ:

$$1/\sigma^2|\alpha_2, \beta_2 \sim Gamma(\alpha_2, \beta_2).$$

Then we could use JAGS for simulation by MCMC with the above model structure.

Results

Data Prepocessing

As the `ratings` and `movies` are very large datasets, we decide to merge these two data frames based on the `movieId` and focus mainly on one specific genre of movies, "Comedy". This not only reduce the total number of observations, but also helps reducing the unexpected correlation between ratings from difference in movie genre. Also, we drop the observation with only 1 total rating as an outlier. We then get a table with movie names corresponding to the **mean**, **sd**, and **n** of its rating.

```
## 'summarise()' has grouped output by 'movieId'. You can override using the
## '.groups' argument.
```

```
## # A tibble: 2,488 x 5
## # Groups:   movieId [2,488]
##    movieId title                                  mean    sd     n
##      <int> <chr>                                 <dbl> <dbl> <int>
##  1       1 Toy Story (1995)                       3.92 0.835   215
##  2       3 Grumpier Old Men (1995)                3.26 1.05     52
##  3       4 Waiting to Exhale (1995)               2.36 0.852     7
##  4       5 Father of the Bride Part II (1995)     3.07 0.907    49
##  5       7 Sabrina (1995)                         3.19 0.978    54
##  6      11 American President, The (1995)         3.67 0.900    70
##  7      12 Dracula: Dead and Loving It (1995)     2.42 1.25     19
##  8      18 Four Rooms (1995)                      3.7  0.909    20
##  9      19 Ace Ventura: When Nature Calls (1995)  2.73 1.06     88
## 10      20 Money Train (1995)                     2.5  0.982    15
## # ... with 2,478 more rows
```

Note that we get the total number of movies as 2488. This is the biggest j as J in our model. And n for each movie, is the biggest i as n_j for the specific movie j.

JAGS Script

Now let's prepare for the JAGS script.

As discussed above, our model will go through $i = 1, \ldots, n_j$ for a Normal distribution of Y_{ij} based on the μ_j with a common shared σ for each j. However, as our Y_{ij} could not split into 2488 vectors with different n_j length for each j, we change our method of iteration.

We go through $y[i]$ from the first rating to the last one. We use N to stand for the total number of ratings, which is 37788.

```r
length(a$rating)
```

```
## [1] 37788
```

We then create a new vector `ID` to store unique IDs from 1 to 2488 to help us identify μ_j. To be clear, the length of `ID` should be the same as N.

```r
length(a$ID)
```

```
## [1] 37788
```

Here is a preview of last 20 rows of data to show how `ID` vector actually looks like.

```
## # A tibble: 20 x 4
## # Groups:   movieId, title [5]
##    movieId title                                     rating    ID
##      <int> <chr>                                      <dbl> <int>
##  1  183897 Isle of Dogs (2018)                          3.5  2484
##  2  184015 When We First Met (2018)                     3.5  2485
##  3  184015 When We First Met (2018)                     3.5  2485
##  4  184791 Fred Armisen: Standup for Drummers (2018)    4    2486
##  5  184791 Fred Armisen: Standup for Drummers (2018)    2.5  2486
##  6  187593 Deadpool 2 (2018)                            4    2487
##  7  187593 Deadpool 2 (2018)                            5    2487
##  8  187593 Deadpool 2 (2018)                            1    2487
##  9  187593 Deadpool 2 (2018)                            4    2487
## 10  187593 Deadpool 2 (2018)                            4.5  2487
## 11  187593 Deadpool 2 (2018)                            2.5  2487
## 12  187593 Deadpool 2 (2018)                            3    2487
## 13  187593 Deadpool 2 (2018)                            5    2487
## 14  187593 Deadpool 2 (2018)                            5    2487
## 15  187593 Deadpool 2 (2018)                            5    2487
## 16  187593 Deadpool 2 (2018)                            4    2487
## 17  187593 Deadpool 2 (2018)                            3.5  2487
## 18  188301 Ant-Man and the Wasp (2018)                  3    2488
## 19  188301 Ant-Man and the Wasp (2018)                  4    2488
## 20  188301 Ant-Man and the Wasp (2018)                  4    2488
```

With this iteration method, we could start compiling our JAGS script.

Following our models, the $y[i]$ follows a normal distribution based on μ_j and common shared σ.

σ follows an inverse gamma distribution based on priors α_1 and β_1.

Each μ_j follows a normal distribution based on common hyperparameters μ and τ.

μ follows a normal distribution based on the second-stage prior μ_0 and s_0.

τ follows an inverse gamma distribution based on second-stage prior α_2 and β_2.

```
model {
  for (i in 1:N) {
    y[i] ~ dnorm(mu_j[ID[i]], invsigmasq)
  }

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

  invsigmasq ~ dgamma(alpha1, beta1)
  sigma <- 1 / sqrt(invsigmasq)
  mu ~ dnorm(mu0, s0)
  invtausq ~ dgamma(alpha2, beta2)
  tau <- 1/ sqrt(invtausq)
}
```

Note that we add more signals `sigma` and `tau` to help track them directly.

List Structure, Prior Parameters and Inits

Now we define data and chose values for prior parameters. Basically we need to choose 6 values, μ_0, s_0, α_1, β_1, α_2, and β_2.

Note that rating for movies is in the ranges from 1 to 5. It is general for us to believe that μ_0 is located around 3. As this is a personal choice and not confident significantly, we could set s_0 as 1.

For τ, we could choose a weakly informative prior with $\alpha_1 = 1$ and $\beta_1 = 1$.

Similarly, we choose a weakly informative prior with $\alpha_2 = 1$ and $\beta_2 = 1$ for σ.

```
y <- a$rating
ID <- a$ID
N <- length(y)
J <- length(unique(ID))
data <- list("y" = y, "ID" = ID,
             "N" = N, "J" = J,
             "mu0" = 3, "s0" = 1,
             "alpha1" = 1, "beta1" = 1,
             "alpha2" = 1, "beta2" = 1)
inits = list(list(mu = 3, invsigmasq = 1, invtausq = 1),
             list(mu = 2.5, invsigmasq = 2, invtausq = 2),
             list(mu = 3.5, invsigmasq = 3, invtausq = 3))
```

Model Simulation

Now let's simulate the model with 3 chain, 1000 adapt.

```
m = jags.model("JAGScode.bug", data, inits, n.chains = 3, n.adapt = 1000)
```

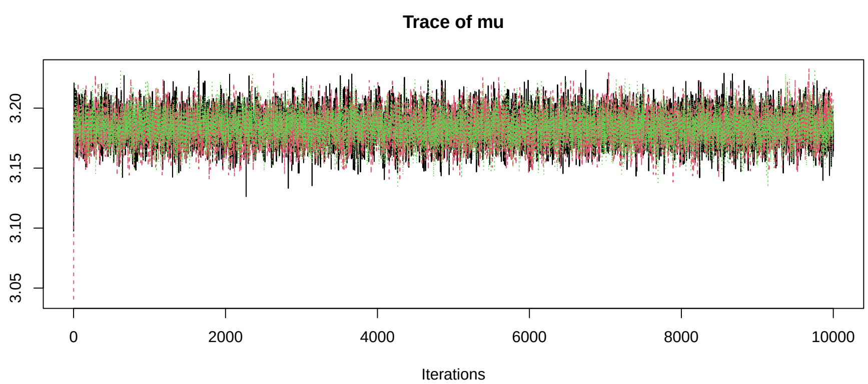
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 37788
##    Unobserved stochastic nodes: 2491
##    Total graph size: 78080
##
## Initializing model
```

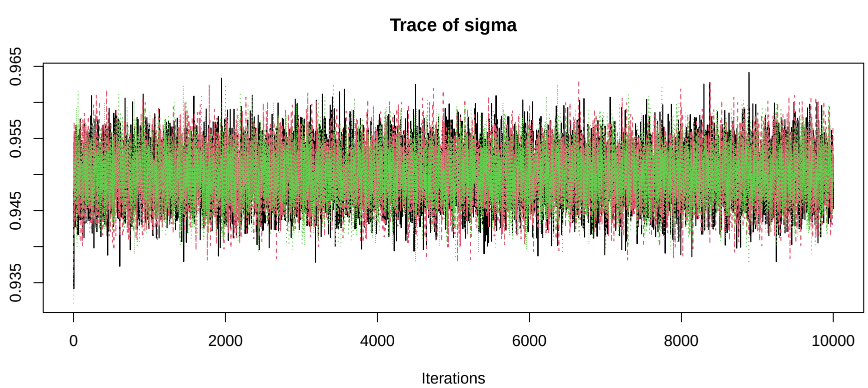
Diagnostics and Summarization

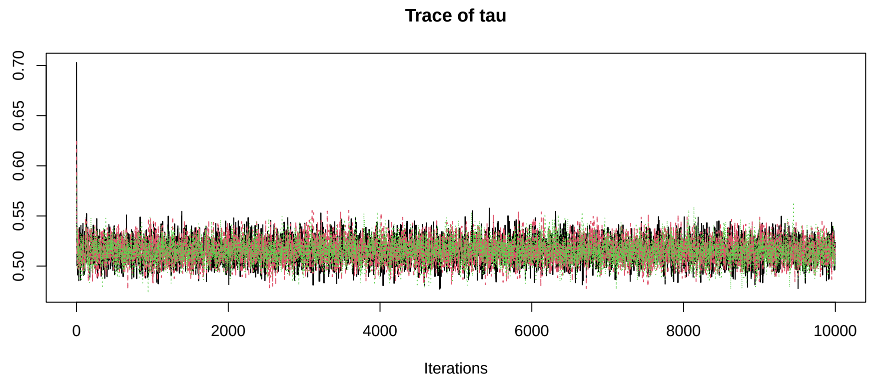
Let's track signals σ, μ, and τ.

```
samples = coda.samples(m, c("sigma", "mu", "tau"), n.iter = 10000)
```

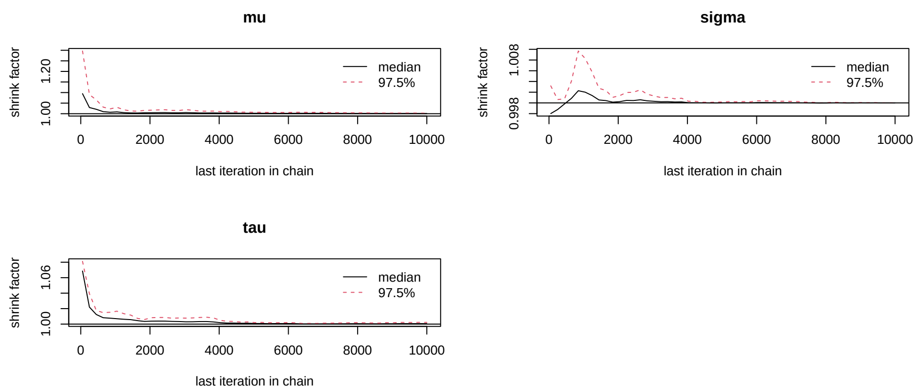
```
traceplot(samples)
```








```r
gelman.plot(samples, autoburnin=FALSE)
```



Note that we confirmed convergence using trace plots and plots of the Gelman statistics for σ, μ, and τ.

The trace plots appeared sufficiently mixed after 1000 iterations or less.

The Gelman statistic plots showed values converging to 1 within a margin of about 1/1000 after 5000 iterations or less.

After 10,000 initial iterations, convergence was confirmed. So, we burn these iterations and sample 100,000 more iterations on each chain.

```r
update(m, 10000)
samples = coda.samples(m, c("sigma", "mu", "tau"), n.iter = 100000)
summary(samples)
```

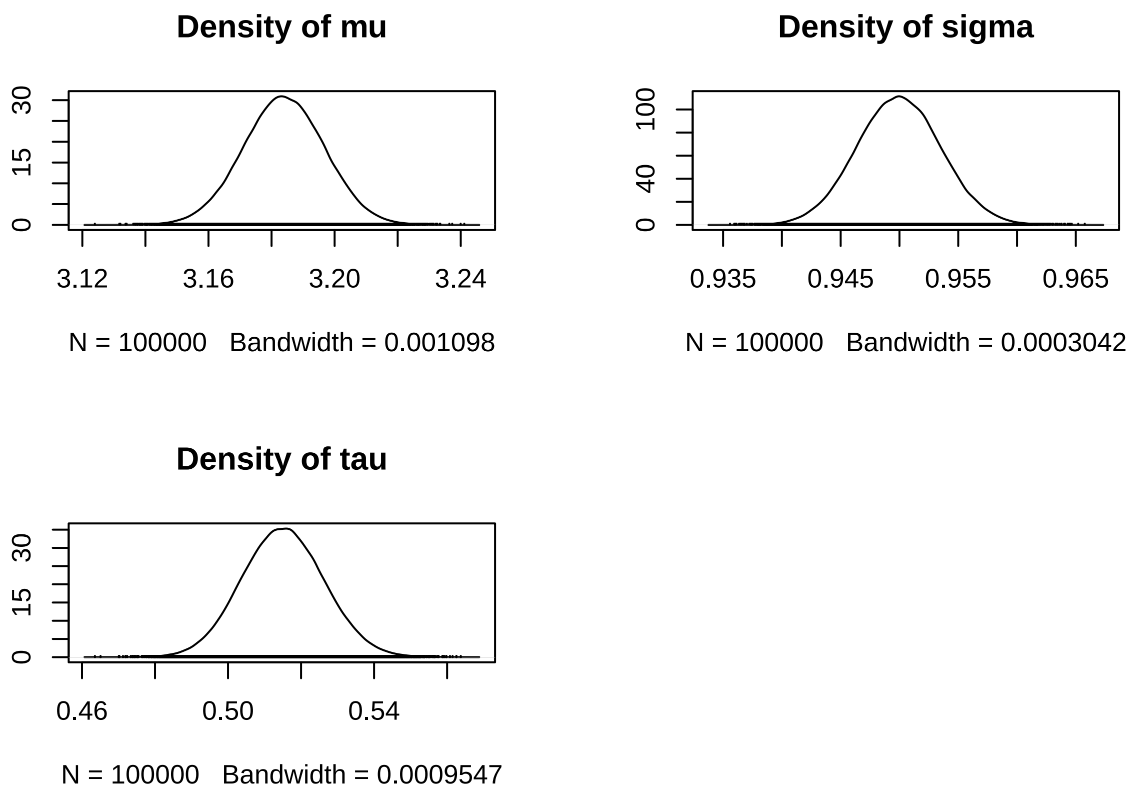
```
##
## Iterations = 20001:120000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 1e+05
##
```

```
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean       SD  Naive SE Time-series SE
## mu    3.1839 0.012907 2.356e-05      3.461e-05
## sigma 0.9500 0.003575 6.528e-06      6.970e-06
## tau   0.5151 0.011220 2.049e-05      3.956e-05
##
## 2. Quantiles for each variable:
##
##         2.5%    25%    50%    75%  97.5%
## mu    3.1585 3.1753 3.1839 3.1926 3.2092
## sigma 0.9430 0.9475 0.9499 0.9524 0.9570
## tau   0.4935 0.5075 0.5150 0.5226 0.5374
```

Note that the Monte Carlo error is less than $\frac{1}{20}$ of SD.

Now let's check the graphical approximations of the posterior densities.

```
plot(samples, trace=FALSE, ask=TRUE)
```



Discussion

In our analyze of film rating, we generate a hierarchical Bayesian model about rating to films of one type, "Comedy." With 3 chain simulation, the sampler convergence after 1000 iterations and final results give us information about posteriors. With 95 intervals, μ is $(3.1587, 3.2092)$, σ is $(0.9430, 0.9570)$, and τ is $(0.4936, 0.5376)$.

Although our model shows convergence and has small MC error, there is still short cuts. As mentioned in the posterior model, we assume that all movies of same type share the σ and each μ_j share the same μ and τ. It is under these assumptions we generate the model. In more general cases, the result might be different. Our limited approach by genre leaves room for more study. Other such potential experiments could model how genre impacts rating, whether by average rating or by how much the ratings for that genre varies. This study certainly suggests that there is much more to be gained from studying similar models that could be created from this dataset.

Contributions

1. Tian Sun: Abstract, methods, and results.
2. Ted Cigler: Introduction, grammar checking, methods, and results.
3. Richard Ma: Discussion, methods, and results.

Reference

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. https://doi.org/10.1145/2827872