STAT 431 Project Report

German Bautista, Eric Bayer, Quentin Wetzel

12/15/2021

Spotify Analysis

Abstract

With the advent of digital music platforms such as Spotify and Apple Music, accessibility to music has ever increased. We were interested in building a hierarchical Bayesian model (HBM) to model the distribution of daily expected stream rates for each song we collected and by genre. With this we can see which songs were popular overall using a Bayesian framework. We used the top 100 most streamed songs on Spotify.

Introduction

Based on our data, the most popular songs on Spotify come from 7 genres: “Dance”, “Electronic”, “Hip -Hop”, “Pop”, “R&B”, “Rap”, and “Rock”. Our goal of this analysis is to determine the distribution of the daily stream rate for the 100 most popular songs on Spotify as well as make some preliminary analyses into the type of genre and song that is most popular on the site. A hierarchical Bayesian model (HBM) built in JAGS is used with a unique λ_{ij} value: which is the expected daily stream rate for song j of genre i. The λ_{ij} value has a hyperprior θ_i based on its genre. Our next section Methods goes more in depth into our dataset and how we built the model. The Appendix section shows our code to perform the hierarchical Bayesian analysis in JAGS as well as the BUGS model. We also interpret various outputs from our model later in the Discussion section. The main contribution of this project is insight into the daily stream rates of the 100 most popular songs on Spotify.

Methods

Our dataset can be found on Kaggle: Spotify Data (https://www.kaggle.com/pavan9065/top-100-most-streamed-songs-on-spotify)

We collected the total number of streams for the top 100 most streamed songs on Spotify via Top 100 Streamed (https://www.thefashionrequest.com/playlist-top-100-most-streamed-songsspotify/) and grouped the songs by genre . However, due to an error in the Kaggle dataset, one observation was deleted due to its low number of streams (lower than the next lowest by a factor of 1000) so our final dataset consisted of 99 total songs. The variable we grouped our songs by in our hierarchical model is genre so we decided to combine similar subgenres in the Kaggle dataset (e.g. combine “Chicago rap” and “dfw rap” into one genre, rap). We settled on seven genres in total: “Dance”, “Electronic”, “Hip -Hop”, “Pop”, “R&B”, “Rap”, and “Rock”. Letting i represent the genre label and j represent the song label, we defined the following variables for $i = 1, 2, \ldots, 7, j = 1, 2, \ldots, 99$:

- Y_{ij}: Number of total streams for song j of genre i (in thousands)
- λ_{ij}: Expected daily stream rate for song j of genre i
- θ_i: A hyperparameter for each genre which λ_{ij} is conditioned on.

The daily stream rate of a song is equal to $\frac{\text{Expected total number of streams}}{\text{Days since release of that song}}$. We used a rate because songs that were released earlier had a longer time to accumulate streams.

We made the following assumptions:

- The number of streams for each song of any genre was independent of the number of streams for each song of a different genre. This means that we did not include the competitiveness between genres; say, accounting an increasing popularity of pop music in a time period due to a decreasing popularity of country music.
- Stream listens occurred independently and could not occur simultaneously
- The average daily stream rate of a genre was constant.

The hierarchical Bayesian model (HBM) had the following layers:

Model: $Y_{ij} | \lambda_{ij} \sim \text{Poisson}(\lambda_{ij})$.

Prior: $\lambda_{ij} | \theta_i \sim^{\text{ind}} \text{Exponential}(\theta_i)$.

Hyperprior: $\theta_i \sim^{\text{iid}} \text{Gamma(a, b)}$.

We chose the Poisson distribution for our model because Y_{ij} represented a count and it met our assumptions. Ideally, we wanted a count model that included correlation between songs and genres as well as allow for two people to listen to a song simultaneously.

We considered three sets of arguments for the hyperprior. Each set was used as a starting point for the three chains in the Markov Chain Monte Carlo (MCMC) setup discussed below. By choosing $a \approx 0, b \approx 0$, we could obtain a proper vague hyperprior so we chose $a = 0.0001, b = 0.0001$. Note that JAGS is not compatible with improper priors so we could not allow a and b to equal zero. To get more informative hyperpriors, we arbitrarily chose $a = 1, b = 1$ and $a = 10, b = 10$.

With our HBM we wanted to perform Markov Chain Monte Carlo (MCMC) in particular Gibbs sampling to give us the posterior distribution $p(\lambda_i, \theta \mid Y_i)$. To do this we used the `rjags` package in R which included the function `coda.samples` to iterate through each of the three chains. After doing so, we verified that the Gelman-Ruben plots and statistics of both λ_ij and θ_i converged and were satisfactory for $i = 1, 2, \cdots 99$, $j = 1, 2 \cdots 7$. We lastly looked at the summary statistics of θ_j for $j = 1, 2, \ldots 7$ and made conclusions about the dataset.

Results

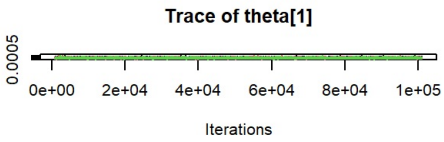
```
## Warning: package 'rjags' was built under R version 4.0.5
```

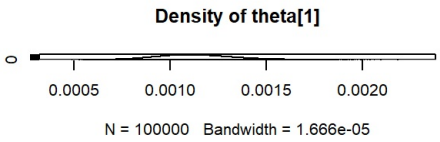
```
## Warning: package 'coda' was built under R version 4.0.5
```

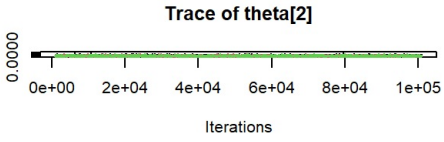
```
## Compiling data graph
##    Resolving undeclared variables
##    Allocating nodes
##    Initializing
##    Reading data back into data table
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 99
##    Unobserved stochastic nodes: 108
##    Total graph size: 217
##
## Initializing model
```

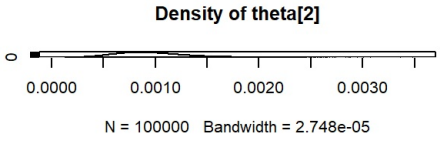
The trace and densities of λ_{ij} were not plotted due to to there being 100 of them. They met all of the convergence requirements that θ_j did.

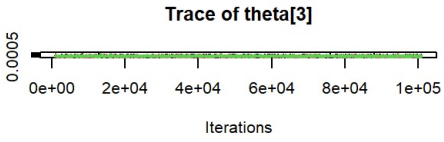
The trace and densities of θ_j for $j = 1, 2, \ldots, 7$ are plotted below.

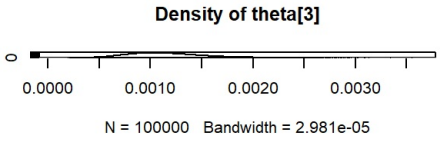


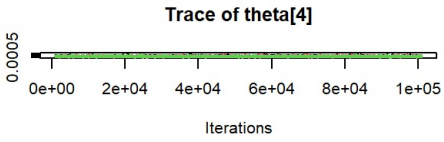


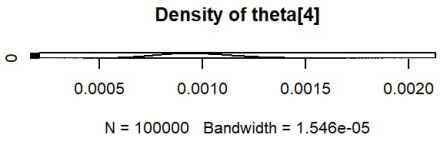


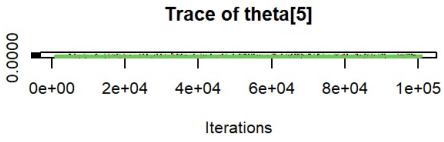


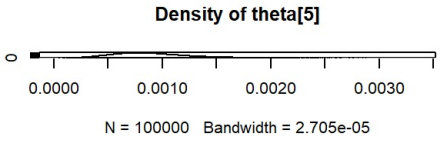


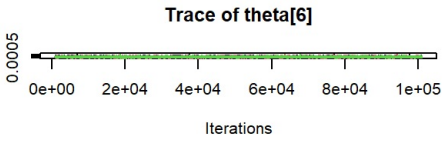


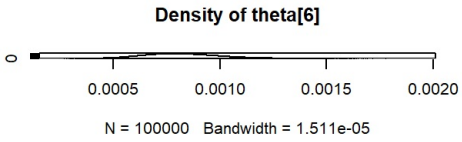


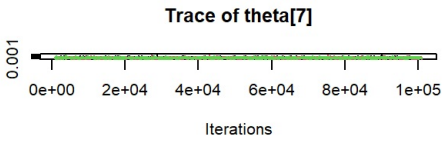


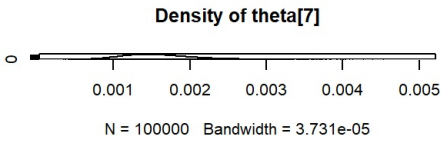




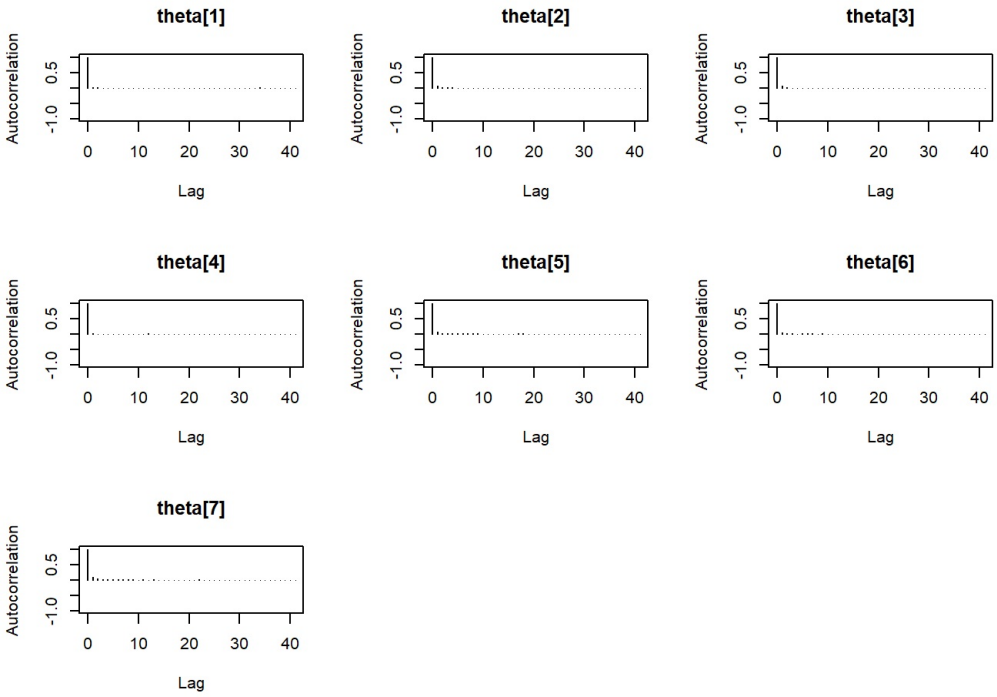








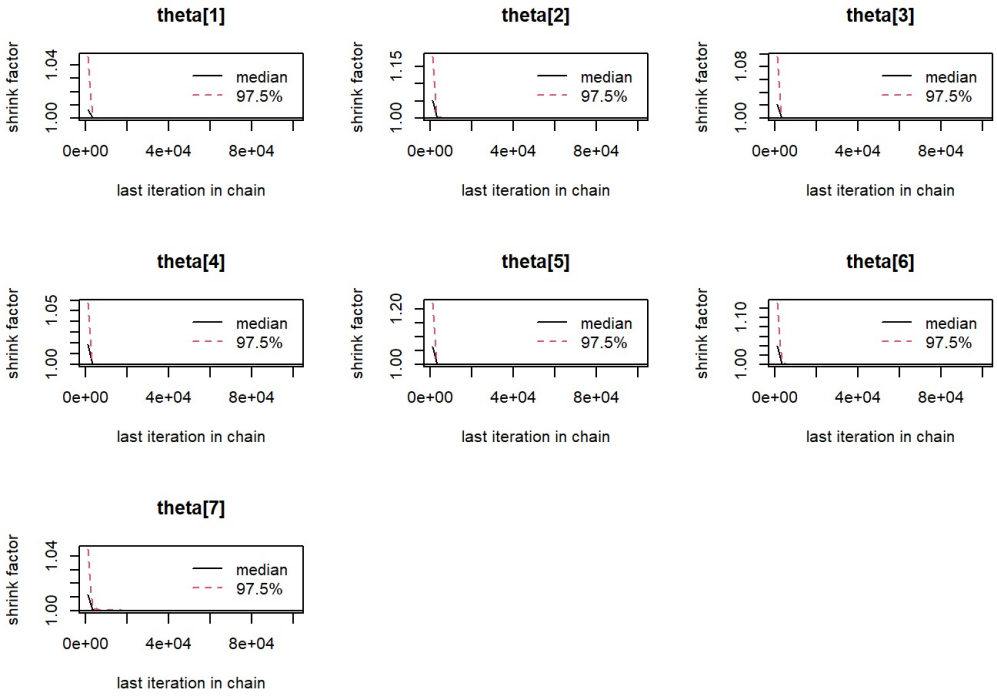
The autocorrelation plots of θ_j for $j = 1, 2, \ldots, 7$ are plotted below.



The Gelman diagnsotics of θ_j for $j = 1, 2, \ldots, 7$ are plotted below.

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## theta[1]          1          1
## theta[2]          1          1
## theta[3]          1          1
## theta[4]          1          1
## theta[5]          1          1
## theta[6]          1          1
## theta[7]          1          1
##
## Multivariate psrf
##
## 1
```

The Gelman plots of θ_j for $j = 1, 2, \ldots, 7$ are plotted below. are below



Summary statistics of θ_j for $j = 1, 2, \ldots, 7$:

```
##
## Iterations = 1001:101000
## 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
## theta[1] 0.0011311 0.0001958 3.575e-07      3.696e-07
## theta[2] 0.0009517 0.0003301 6.026e-07      6.797e-07
## theta[3] 0.0011407 0.0003553 6.487e-07      6.977e-07
## theta[4] 0.0009699 0.0001818 3.319e-07      3.472e-07
## theta[5] 0.0008792 0.0003239 5.913e-07      7.343e-07
## theta[6] 0.0008221 0.0001776 3.243e-07      3.688e-07
## theta[7] 0.0015784 0.0004476 8.172e-07      1.019e-06
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%    97.5%
## theta[1] 0.0007798 0.0009937 0.0011200 0.0012560 0.001547
## theta[2] 0.0004132 0.0007151 0.0009157 0.0011478 0.001697
## theta[3] 0.0005546 0.0008853 0.0011042 0.0013547 0.001937
## theta[4] 0.0006464 0.0008423 0.0009583 0.0010858 0.001358
## theta[5] 0.0003532 0.0006466 0.0008430 0.0010725 0.001610
## theta[6] 0.0005103 0.0006967 0.0008095 0.0009345 0.001204
## theta[7] 0.0008434 0.0012580 0.0015316 0.0018456 0.002582
```

Discussion

With 100,000 iterations and 3 chains, each λ_{ij} and θ_i converged according to their trace plots. The Gelman-Ruben statistic and the autocorrelation plots satisfied their criterias, the Gelman-Ruben statistic was near 1 for all λ_{ij} and θ_i values, and the autocorrelation plot showed little to no autocorrelation. The time-series standard deviation is also well below 1/20 of the standard deviations for each statistic. With all the diagnostics showing the chains are satisfactory, we can make various posterior credible intervals on streams rates for different songs and other parameters. For example, $\lambda_{1,1}$ is the first song in our first genre and it is called “Closer” by the Chainsmokers and classified as “Dance” for its genre. A 95 percent posterior credible interval for this song is [1088,1133] which means there is a 95 percent chance that the expected daily stream rate (in thousands) for the song “Closer” by the Chainsmokers is between 1088 and 1133.

The analysis done here can show which songs within the top 100 list are more likely to be streamed than others and it can show statistics on the seven genres included in our model. However, our scope is limited to the 99 songs used in the dataset.

With more time, we would like to create a model where we could pool songs within each genre and get better statistics on the overall genres. This would give us a better idea of which genres have the highest stream rate and useful posterior credible intervals. We could also analyze daily stream rates over time (most likely as days) as a time series to see how a song’s daily stream rate changes over time. We can then apply Bayesian methods to the time series.

Ideally, we would like a larger dataset to work with as well. Our conclusions only apply to the 99 songs that we looked at, and even though they had a very high amount of streams, it is still a small sample of all of the music streamed on Spotify.

The dataset and our code can be found on Github: spotify-snalysis (https://github.com/gbautista365/spotify-analysis)

Contributions

Honestly, we all contributed what we could and we were always in touch throughout the project. We gave it our best effort on this project. This is a very good team. Believe me.

German Bautista (1/3): Brainstorming, Revised paper, Attended meetings, Code

Eric Bayer (1/3): Brainstorming, Revised paper, Attended meetings, Code

Quentin Wetzel (1/3): Brainstorming, Revised paper, Attended meetings, Code

Appendix

```
# Load library
library(rjags)

# Read data
data = read.table("ungrouped_spotify_data.txt")

# Make the streams per day variable.
data$streams_day = data$stream / data$days

# This orders alphabetically by genre
# This is vital for our bugs model to makes sense
data = data[order(data$top.genre), ]

# This resets the row numbers so again the bugs model works
rownames(data) = NULL

# Shows how the indices were selected to use in the bugs file
indicator = c(match(unique(data$top.genre), data$top.genre), length(data[,1])+1)

# Add thousands of streams per day to Y
d = list(Y = data[ , 4])

# Set arbitrary uninformative priors
# Feel free to change a, b for testing purposes
inits = list(list(a = .0001, b = .0001),
             list(a = 1000, b = 1000),
             list(a = .0001, b = 1000))

#Run the JAGS model
m = jags.model("hierarchical4.bug", d, inits, n.chains=3)

# Includes lambda for convergence checking
x = coda.samples(m, c("theta", "lambda"), n.iter=1e5)

# In paper, only plot the theta graphs
z = coda.samples(m, c("theta"), n.iter = 1e5)

### Assess convergence

plot(x, smooth = FALSE, ask = TRUE)

autocorr.plot(x[1], ask = TRUE)

gelman.diag(x, autoburnin = FALSE)

gelman.plot(x, autoburnin = FALSE, ask = TRUE)

# Check statistics after burn-in
# Verify that Time-series SE less than 1/20 of SD
summary(x)
```

```
# hierarchical4.bug file used for HBM
data {
  n.genres = 7
  ind = c(1, 31, 36, 43, 68, 72, 90, 100)
}
model {
  for (i in 1:n.genres) {
    for (j in ind[i]:(ind[i + 1] - 1)) {
      Y[j] ~ dpois(lambda[j])
      lambda[j] ~ dexp(theta[i])
    }
    theta[i] ~ dgamma(a,b)
  }
  a ~ dexp(0.001)
  b ~ dexp(0.001)
}
```