

Slide 16 - R Output

Elvan Ceyhan

11/25/2024

```
#Set Working Directory to Source File Location
library("rstudioapi") # Load rstudioapi package
#setwd(dirname(getActiveDocumentContext()$path)) # Set working directory to source file location
#getwd()
```

Some required packages:

```
# Load packages
library(rstan)
library(ggplot2)
library(bayesrules) # Bayesian methods
library(tidyverse) # Data manipulation and visualization
library(rstanarm) # Bayesian regression
library(bayesplot) # Visualization for Bayesian models
library(tidybayes) # Tools for working with Bayesian model outputs
library(broom.mixed) # Tidying mixed models
library(forcats) # Factor handling
```

The Spotify Dataset:

This script analyzes a subset of Spotify data, focusing on song popularity and artist-level summaries.

```
# Load the Spotify dataset
data(spotify)

#Print the first 10 rows of the data set
cat("First 10 rows of the Spotify dataset: \n")
```

First 10 rows of the Spotify dataset:

```
print(spotify[1:10, ])
```

```
## # A tibble: 10 x 23
##   track_id      title artist popularity album_id album_name album_release_date
##   <chr>         <chr> <fct>         <dbl> <chr>      <chr>      <chr>
## 1 7sYAS4CpbV90o~ On &~ Alok          79 2a9AGiv~ On & On    2019-11-22
## 2 1YSwQvw1NrEPf~ All ~ Alok          56 2FdyKqN~ All The L~ 2019-07-19
## 3 39cmB3ZoTOLw0~ Hear~ Alok          75 6fpsA5a~ Hear Me N~ 2016
## 4 2Dnb6yPGUq0vm~ The ~ Alok          65 0hH9tb1~ The Wall   2019-06-28
## 5 1a5Yu5L18qNxV~ Hear~ Alok          52 6apmeAA~ Hear Me N~ 2016-01-01
## 6 74RrMiuS0zAHZ~ Toda~ Alok          45 3GQZqxm~ Toda La N~ 2018-05-25
## 7 7sYAS4CpbV90o~ On &~ Alok          79 2a9AGiv~ On & On    2019-11-22
## 8 0Qp1IH1EkRCNa~ The ~ Alok          61 4w68G9o~ The Wall   2019-06-28
## 9 36C0Gw2RmYWy5~ Tell~ Alok          61 1R2C7EG~ Tell Me W~ 2019-08-30
```

```
## 10 1UTXQrmAL4ola~ Meta~ Alok          61 2ds371e~ Metaphor    2019-02-01
## # i 16 more variables: playlist_name <chr>, playlist_id <chr>, genre <chr>,
## #   subgenre <chr>, danceability <dbl>, energy <dbl>, key <dbl>,
## #   loudness <dbl>, mode <dbl>, speechiness <dbl>, acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
## #   duration_ms <dbl>
```

```
# Select relevant variables and reorder artists by mean popularity
```

```
spotify <- spotify %>%
  select(artist, title, popularity) %>%
  mutate(artist = fct_reorder(artist, popularity, .fun = mean))
```

```
# Display the first three rows of the modified dataset
```

```
cat("First three rows of the modified dataset:\n")
```

```
## First three rows of the modified dataset:
```

```
head(spotify, 3)
```

```
## # A tibble: 3 x 3
##   artist title      popularity
##   <fct> <chr>      <dbl>
## 1 Alok   On & On         79
## 2 Alok   All The Lies    56
## 3 Alok   Hear Me Now     75
```

```
# Summary of dataset
```

```
cat("Total number of songs in the dataset:", nrow(spotify), "\n")
```

```
## Total number of songs in the dataset: 350
```

```
cat("Total number of unique artists in the dataset:", nlevels(spotify$artist), "\n")
```

```
## Total number of unique artists in the dataset: 44
```

```
# List of unique artists
```

```
cat("Artists in the dataset:\n")
```

```
## Artists in the dataset:
```

```
levels(spotify$artist)
```

```
## [1] "Mia X"          "Chris Goldarg"    "Soul&Roll"
## [4] "Honeywagon"     "Röyksopp"         "Freestyle"
## [7] "DA Image"       "Jean Juan"        "TV Noise"
## [10] "Kid Frost"      "Tamar Braxton"    "BUNT."
## [13] "MTK"           "Atlas Genius"     "Jazzinuf"
## [16] "Elisa"          "House Of Pain"    "Black Stone Cherry"
## [19] "C-Kan"          "Zeds Dead"        "David Lee Roth"
## [22] "NODE"           "Michael Kiwanuka" "The Wrecks"
## [25] "The xx"         "Placebo"          "Mike WiLL Made-It"
## [28] "Kendrick Lamar" "X Ambassadors"    "Hinder"
## [31] "Au/Ra"          "Missy Elliott"    "The Blaze"
## [34] "Vampire Weekend" "Alok"             "León Larregui"
## [37] "Sufjan Stevens" "Beyoncé"          "Frank Ocean"
## [40] "Sean Kingston"  "J. Cole"          "Camila Cabello"
## [43] "Lil Skies"      "Camilo"
```

```

# Group songs by artist and compute summary statistics (count and mean popularity)
artist_means <- spotify %>%
  group_by(artist) %>%
  summarize(
    count = n(), # Number of songs per artist
    mean_popularity = mean(popularity) # Mean popularity per artist
  )

# Display the first two and last two artists based on reordering
cat("First two and last two artists by mean popularity:\n")

## First two and last two artists by mean popularity:
artist_means %>% slice(1:2, 43:44)

## # A tibble: 4 x 3
##   artist      count mean_popularity
##   <fct>      <int>      <dbl>
## 1 Mia X         4         13.2
## 2 Chris Goldarg 10         16.4
## 3 Lil Skies      3         79.3
## 4 Camilo        9         81

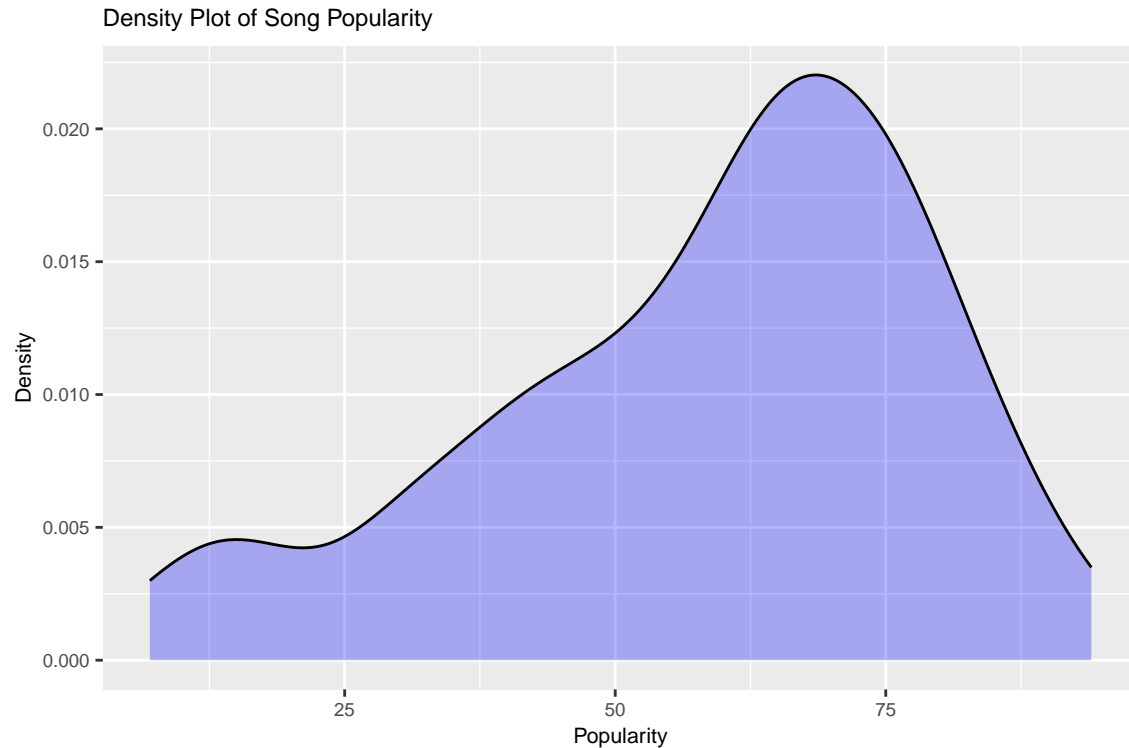
# Range of the number of songs per artist
cat("Range of songs per artist:\n")

## Range of songs per artist:
artist_means %>%
  summarize(min_songs = min(count), max_songs = max(count))

## # A tibble: 1 x 2
##   min_songs max_songs
##   <int>      <int>
## 1         2        40

# Density plot of song popularity across all artists
ggplot(spotify, aes(x = popularity)) +
  geom_density(fill = "blue", alpha = 0.3) + # Add color and transparency for better visualization
  labs(
    title = "Density Plot of Song Popularity",
    x = "Popularity", y = "Density"
  )

```



Fitting an Intercept-Only Regression Model:

This section fits a Bayesian regression model with only an intercept to analyze the overall mean popularity, summarizes the results, and compares posterior predictions to observed artist-level means.

Fit an intercept-only Bayesian regression model

```
spotify_complete_pooled <- stan_glm(
  popularity ~ 1, # Intercept-only model
  data = spotify,
  family = gaussian,
  prior_intercept = normal(50, 2.5, autoscale = TRUE), # Informative prior for the intercept
  prior_aux = exponential(1, autoscale = TRUE), # Prior for auxiliary parameters
  chains = 4, iter = 10000, seed = 12345 # 4 chains, 10,000 iterations
)
```

Summarize posterior distributions for intercept and auxiliary parameters

```
complete_summary <- tidy(
  spotify_complete_pooled,
  effects = c("fixed", "aux"),
  conf.int = TRUE, conf.level = 0.90 # 90% credible intervals
)
cat("Summary of the Intercept-Only Model:\n")
```

```
## Summary of the Intercept-Only Model:
```

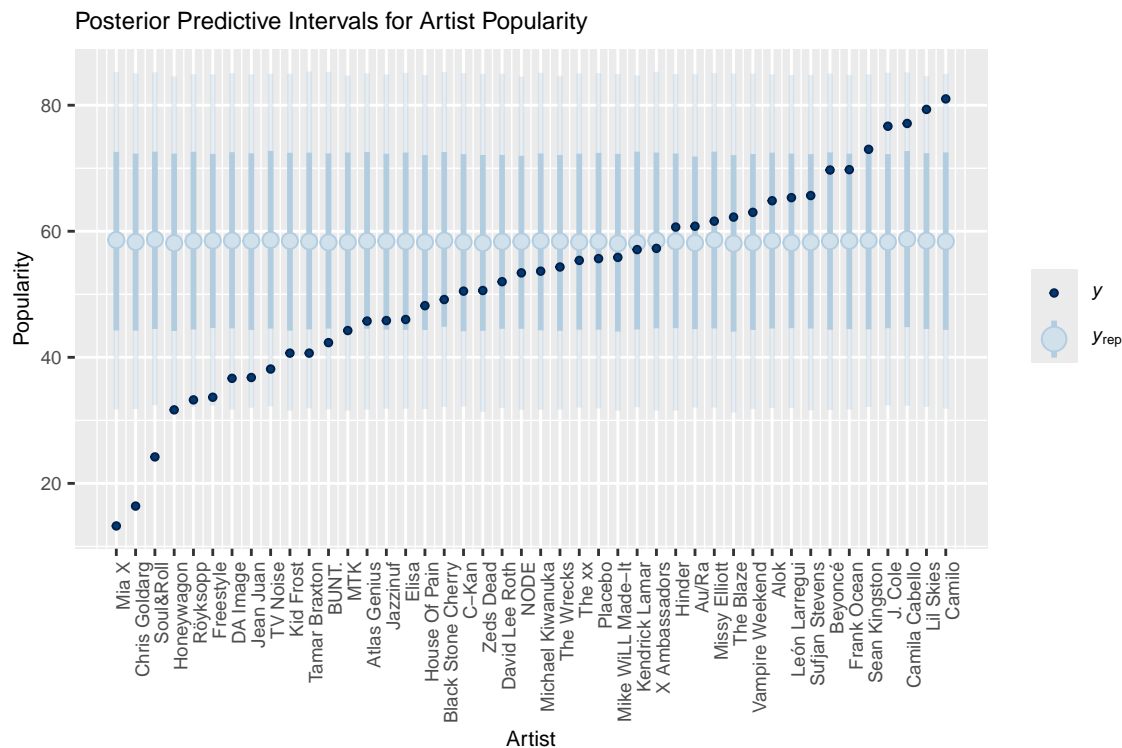
```
print(complete_summary)
```

```
## # A tibble: 3 x 5
##   term      estimate std.error conf.low conf.high
##   <chr>      <dbl>    <dbl>   <dbl>   <dbl>
## 1 (Intercept)  58.4      1.10    56.5    60.2
## 2 sigma       20.6      0.783   19.4    22.0
## 3 mean_PPD    58.4      1.57    55.8    61.0

# Generate posterior predictive means for artist-level popularity
predictions_complete <- posterior_predict(
  spotify_complete_pooled,
  newdata = artist_means # Predict for artist means
)

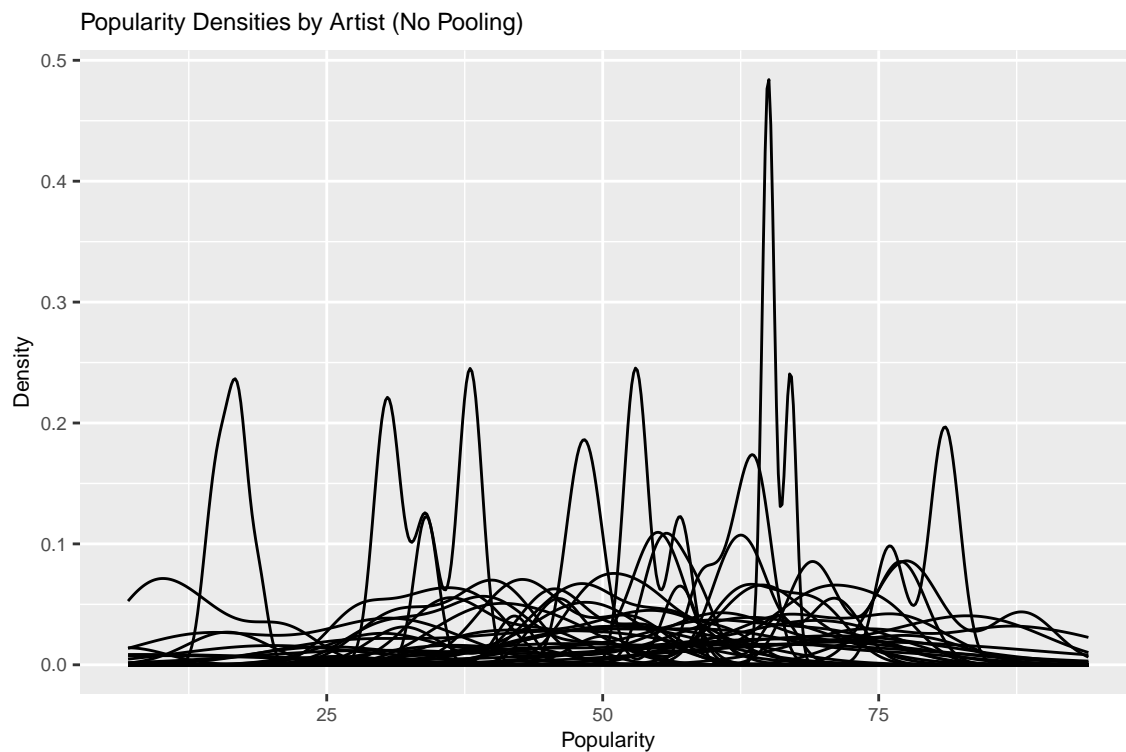
# Visualize posterior predictive intervals with actual artist popularity
ppc_intervals(
  y = artist_means$mean_popularity, # Observed artist mean popularity
  yrep = predictions_complete, # Posterior predictive samples
  prob_out = 0.80 # 80% prediction intervals
) +
  ggplot2::scale_x_continuous(
    labels = artist_means$artist, # Artist names as x-axis labels
    breaks = 1:nrow(artist_means) # Match labels with artist positions
  ) +
  ggplot2::theme(axis.text.x = element_text(angle = 90, hjust = 1)) + # Rotate x-axis labels
  labs(
    title = "Posterior Predictive Intervals for Artist Popularity",
    x = "Artist",
    y = "Popularity"
  )

```



No Pooling and Hierarchical Modeling of Spotify Popularity Data

```
# No pooling: Separate popularity densities for each artist
ggplot(spotify, aes(x = popularity, group = artist)) +
  geom_density() +
  labs(
    title = "Popularity Densities by Artist (No Pooling)",
    x = "Popularity", y = "Density"
  )
```



```
# No pooling model estimation: Fit a separate intercept for each artist
spotify_no_pooled <- stan_glm(
  popularity ~ artist - 1, # Separate intercepts for each artist
  data = spotify,
  family = gaussian,
  prior = normal(50, 2.5, autoscale = TRUE),
  prior_aux = exponential(1, autoscale = TRUE),
  chains = 4, iter = 10000, seed = 12345
)

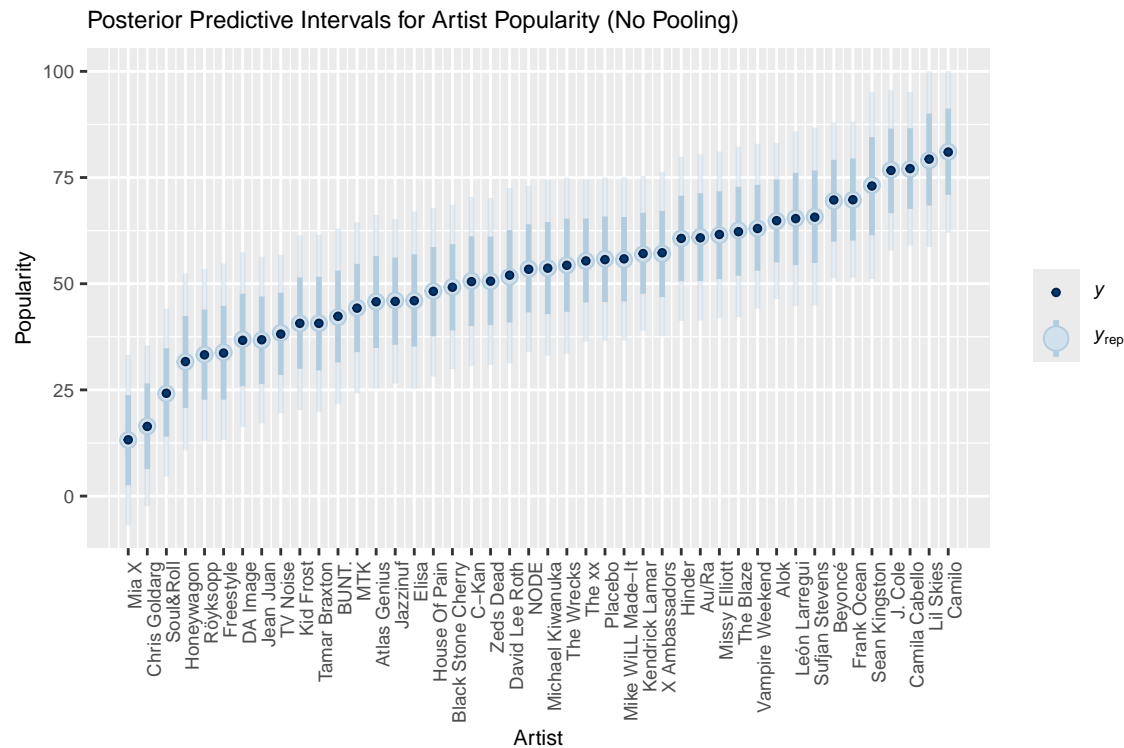
# Posterior predictive distributions: Simulate the posterior predictive values
predictions_no <- posterior_predict(
  spotify_no_pooled,
  newdata = artist_means
)

# Plot posterior predictive intervals for no pooling model
ppc_intervals(
  y = artist_means$mean_popularity, # Observed artist-level mean popularity
  yrep = predictions_no, # Posterior predictive samples
)
```

```

prob_out = 0.80 # 80% prediction intervals
) +
ggplot2::scale_x_continuous(
  labels = artist_means$artist, # Artist names on x-axis
  breaks = 1:nrow(artist_means)
) +
ggplot2::theme(axis.text.x = element_text(angle = 90, hjust = 1)) + # Rotate x-axis labels
labs(
  title = "Posterior Predictive Intervals for Artist Popularity (No Pooling)",
  x = "Artist",
  y = "Popularity"
)

```

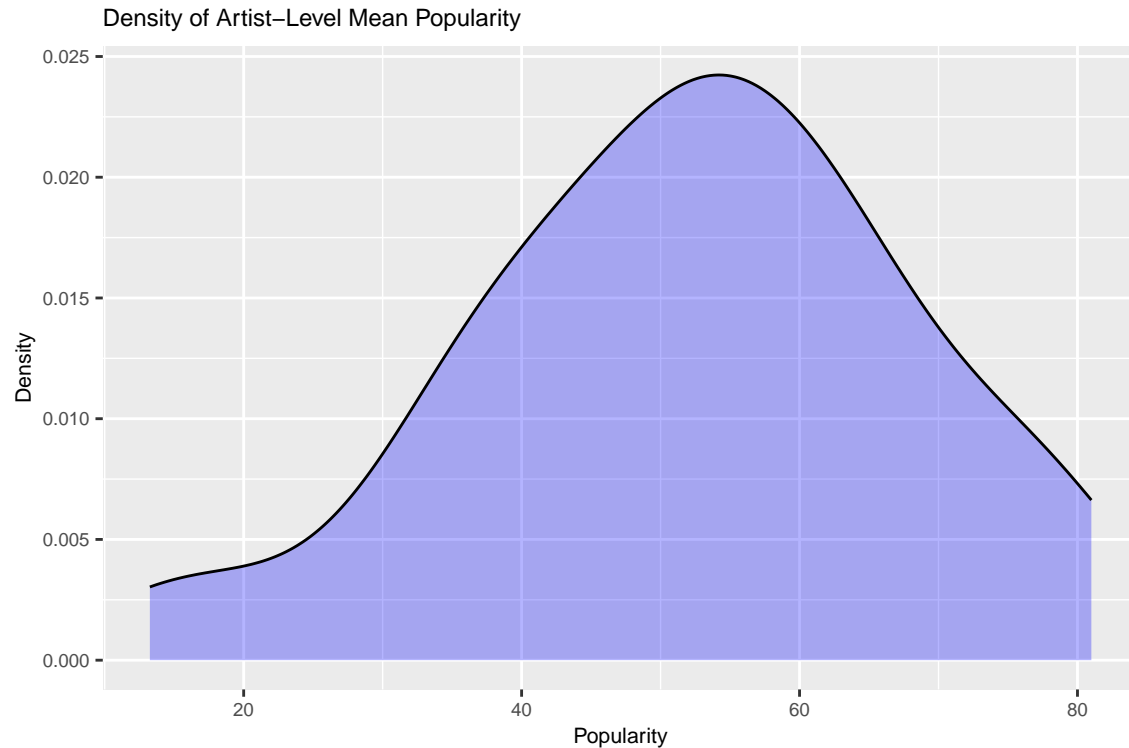


Hierarchical Modeling: Accounting for artist-specific variability

```

# Distribution of artist-level sample means
ggplot(artist_means, aes(x = mean_popularity)) +
  geom_density(fill = "blue", alpha = 0.3) +
  labs(
    title = "Density of Artist-Level Mean Popularity",
    x = "Popularity", y = "Density"
  )

```



```
# Fit a hierarchical model with varying intercepts for each artist
spotify_hierarchical <- stan_glmer(
  popularity ~ (1 | artist), # Random intercept for each artist
  data = spotify,
  family = gaussian,
  prior_intercept = normal(50, 2.5, autoscale = TRUE), # Prior for global intercept
  prior_aux = exponential(1, autoscale = TRUE), # Prior for auxiliary parameter
  prior_covariance = decov(reg = 1, conc = 1, shape = 1, scale = 1), # Prior for random effects
  chains = 4, iter = 10000, seed = 12345
)
```

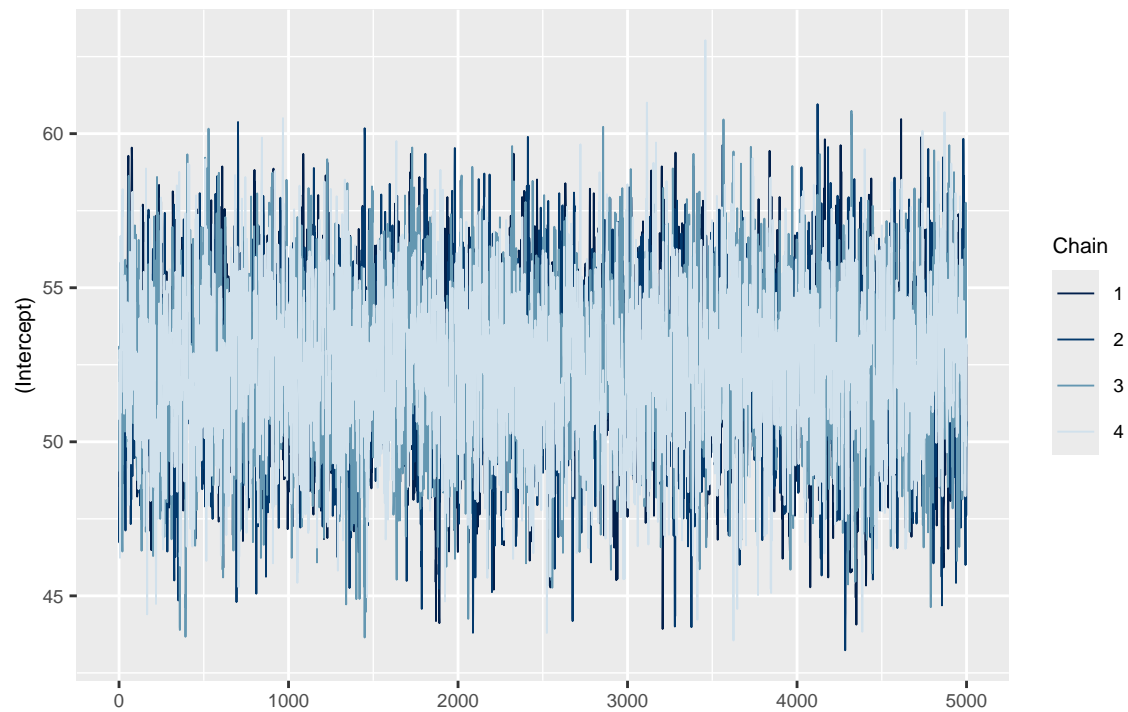
MCMC Diagnostics and Posterior Analysis for Spotify Hierarchical Model

```
# MCMC Diagnostics: Trace plots and posterior densities

# There are lots of parameters here, so there are a bunch of plots here:
#mcmc_trace(spotify_hierarchical) # don't run, takes a lot of time to show 45+ trace plots ...
#mcmc_dens_overlay(spotify_hierarchical)

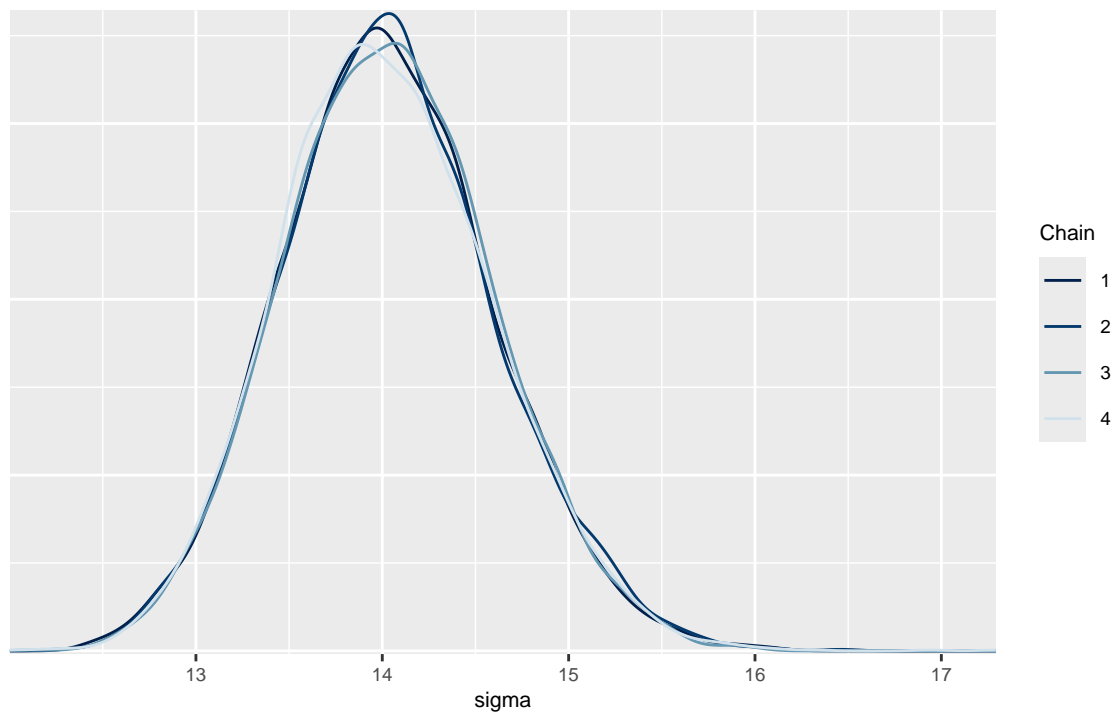
# Trace plot for the global intercept
mcmc_trace(spotify_hierarchical, pars = '(Intercept)') +
  ggtitle("Trace Plot for Global Intercept (Mu)")
```


Trace Plot for Global Intercept (Mu)



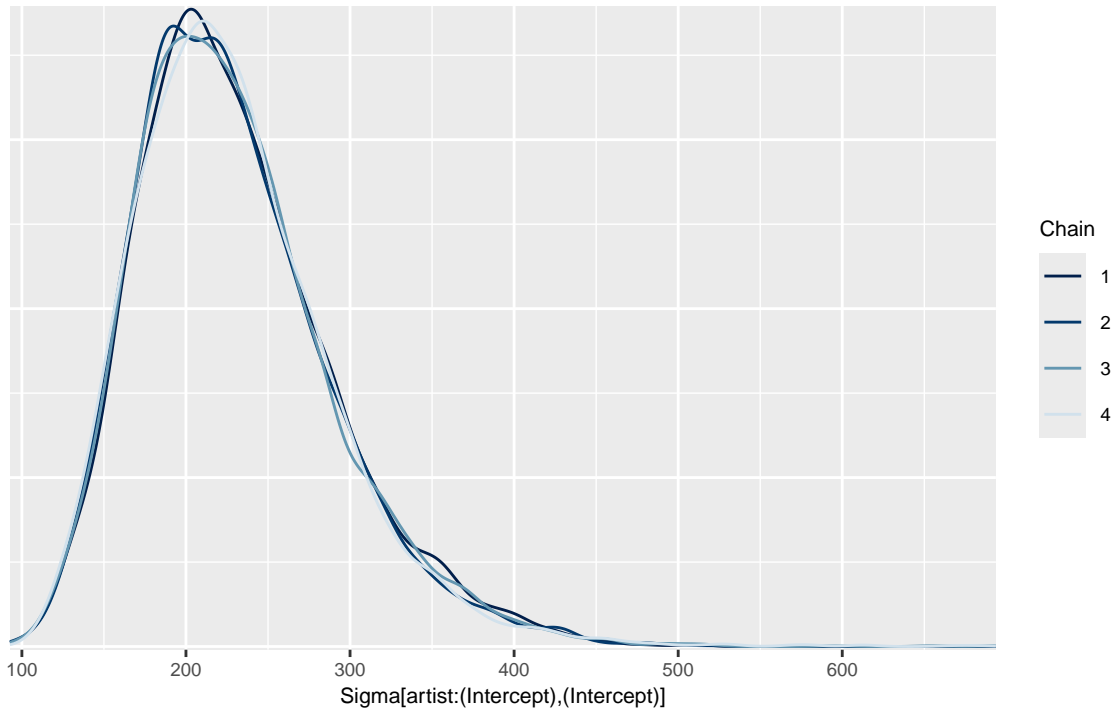
```
# Posterior density plot for sigma (residual standard deviation)
mcmc_dens_overlay(spotify_hierarchical, pars = 'sigma') +
  ggtitle("Posterior Density for Residual Standard Deviation (Sigma)")
```

Posterior Density for Residual Standard Deviation (Sigma)



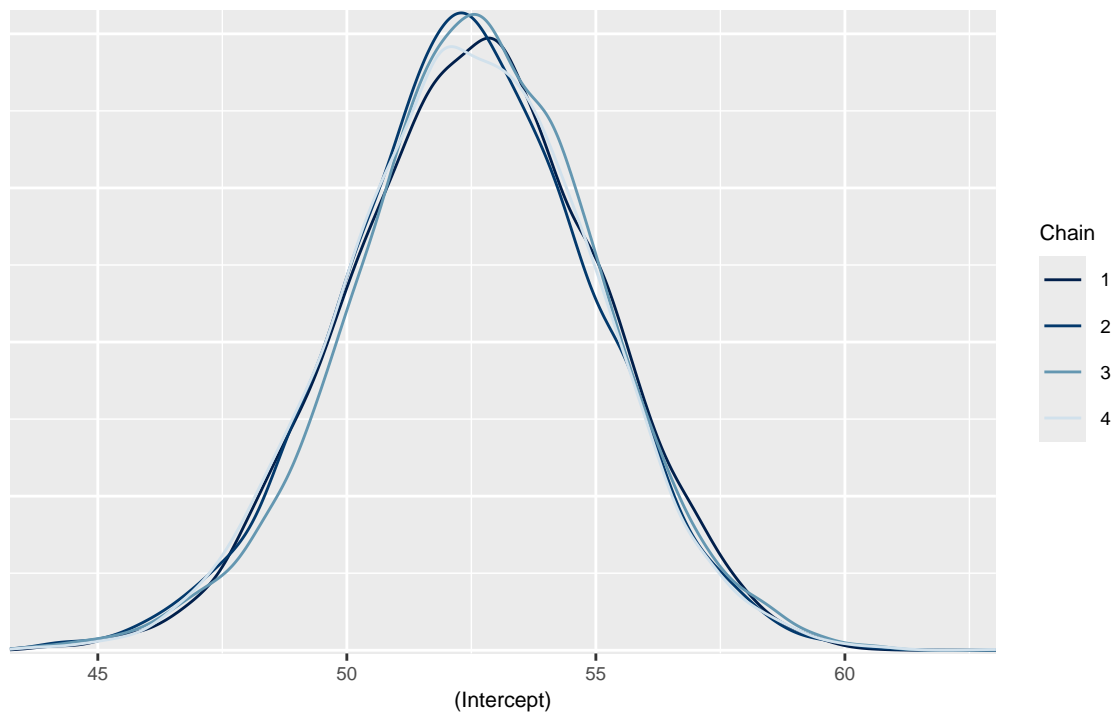
```
# Posterior density for sigma_mu (random effect variability)
mcmc_dens_overlay(spotify_hierarchical, pars = 'Sigma[artist:(Intercept),(Intercept)]') +
  ggtitle("Posterior Density for Artist-Level Variability (Sigma_mu)")
```

Posterior Density for Artist-Level Variability (Sigma_mu)



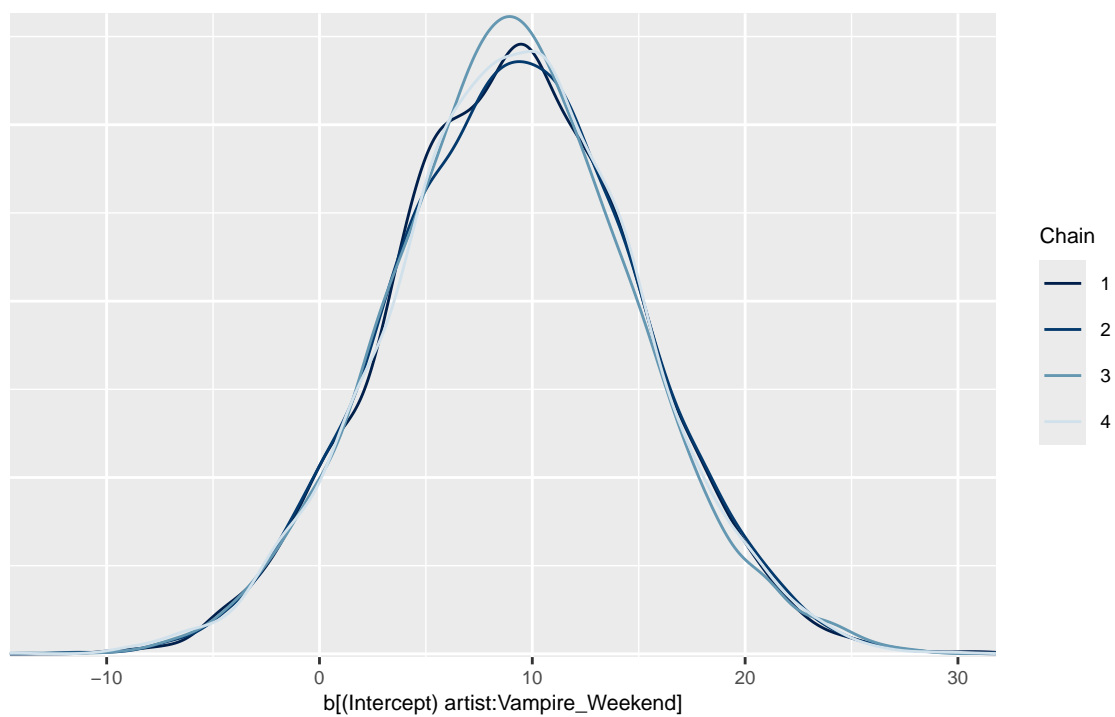
```
# Posterior density for the global intercept (Mu)
mcmc_dens_overlay(spotify_hierarchical, pars = '(Intercept)') +
  ggtitle("Posterior Density for Global Intercept (Mu)")
```

Posterior Density for Global Intercept (μ)

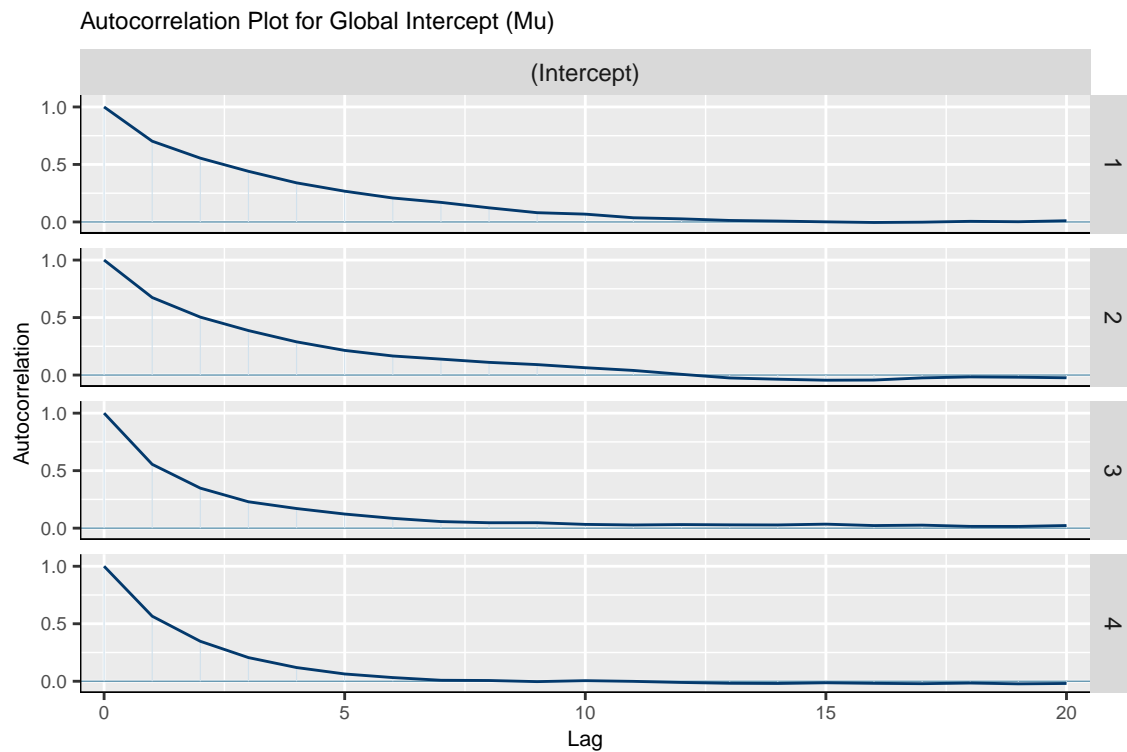


```
# Posterior density for a specific artist's random effect (e.g., Vampire Weekend)
mcmc_dens_overlay(spotify_hierarchical, regex_pars = "Vampire_Weekend") +
  ggtitle("Posterior Density for Vampire Weekend's Effect")
```

Posterior Density for Vampire Weekend's Effect



```
# Autocorrelation Function (ACF) plot for the global intercept
mcmc_acf(spotify_hierarchical, pars = '(Intercept)') +
  ggtitle("Autocorrelation Plot for Global Intercept (Mu)")
```



Posterior Summaries

```
# Posterior summary for the global intercept (Mu)
global_mu_summary <- tidy(
  spotify_hierarchical,
  effects = "fixed",
  conf.int = TRUE,
  conf.level = 0.80
)
```

```
cat("Posterior Summary for Global Mu:\n")
```

```
## Posterior Summary for Global Mu:
```

```
print(global_mu_summary)
```

```
## # A tibble: 1 x 5
##   term      estimate std.error conf.low conf.high
##   <chr>      <dbl>    <dbl>   <dbl>   <dbl>
## 1 (Intercept)  52.5      2.47    49.3    55.7
```

```
# Posterior summary for sigma_y (residual standard deviation) and sigma_mu (random effect variability)
random_effects_summary <- tidy(
  spotify_hierarchical,
  effects = "ran_pars"
```

```
)
cat("Posterior Summary for Sigma_y and Sigma_mu:\n")
```

```
## Posterior Summary for Sigma_y and Sigma_mu:
```

```
print(random_effects_summary)
```

```
## # A tibble: 2 x 3
##   term                group estimate
##   <chr>              <chr>     <dbl>
## 1 sd_(Intercept).artist artist      15.2
## 2 sd_Observation.Residual Residual    14.0
```

Artist-Level Posterior Summaries

```
# Extract MCMC chains for artist-specific intercepts (mu_j)
artist_chains <- spotify_hierarchical %>%
  spread_draws(`(Intercept)`, b[,artist]) %>%
  mutate(mu_j = `(Intercept)` + b) # Combine global intercept with artist-specific effects

# Summarize mu_j with 80% credible intervals and reorder by mean mu_j
artist_summary_scaled <- artist_chains %>%
  select(`(Intercept)`, -b) %>%
  mean_qi(.width = 0.80) %>%
  mutate(artist = fct_reorder(artist, mu_j))

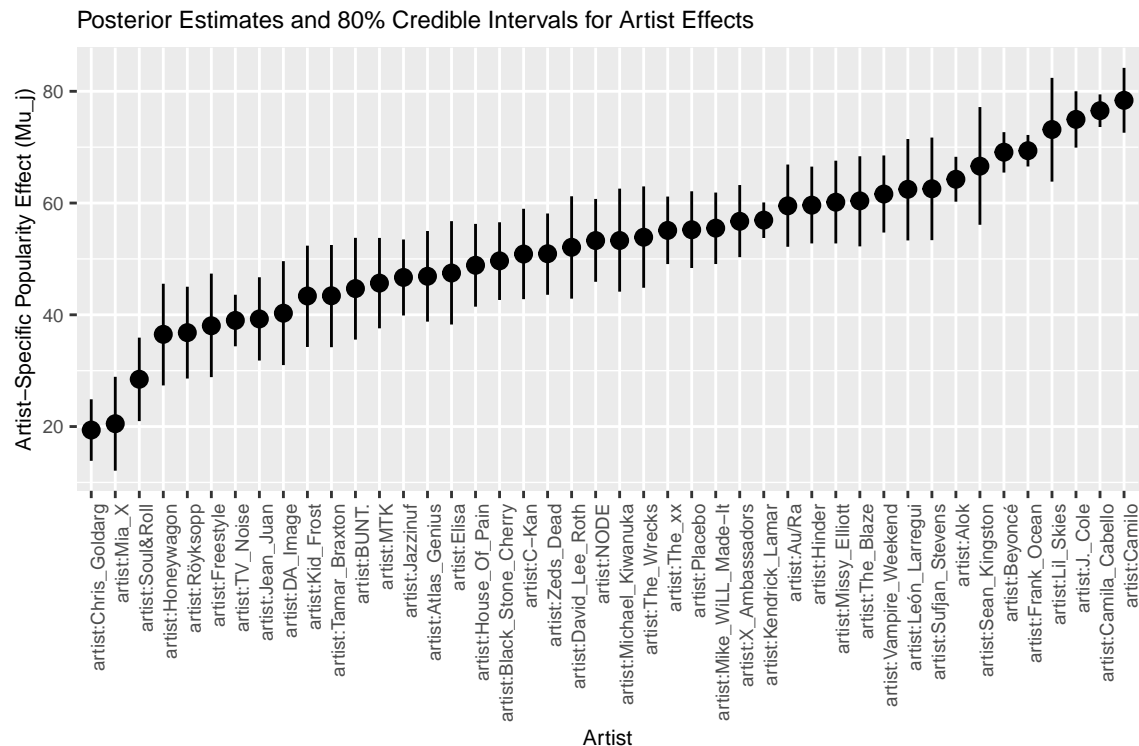
# Display a sample of the summarized results
cat("Artist-Level Posterior Summaries (Top Rows):\n")
```

```
## Artist-Level Posterior Summaries (Top Rows):
```

```
print(artist_summary_scaled %>%
  select(artist, mu_j, .lower, .upper) %>%
  head(4))
```

```
## # A tibble: 4 x 4
##   artist                mu_j .lower .upper
##   <fct>              <dbl>  <dbl>  <dbl>
## 1 artist:Alok          64.3   60.2   68.3
## 2 artist:Atlas_Genius  46.9   38.8   55.0
## 3 artist:Au/Ra         59.5   52.2   66.9
## 4 artist:Beyoncé       69.1   65.5   72.7
```

```
# Plot estimates and 80% credible intervals for mu_j
ggplot(artist_summary_scaled, aes(x = artist, y = mu_j, ymin = .lower, ymax = .upper)) +
  geom_pointrange() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(
    title = "Posterior Estimates and 80% Credible Intervals for Artist Effects",
    x = "Artist",
    y = "Artist-Specific Popularity Effect (Mu_j)"
  )
```



```
# Checking artists with wide and narrow credible intervals
artist_means %>%
  filter(artist %in% c("Frank Ocean", "Lil Skies"))
```

```
## # A tibble: 2 x 3
##   artist      count mean_popularity
##   <fct>      <int>      <dbl>
## 1 Frank Ocean    40        69.8
## 2 Lil Skies      3        79.3
```

```
# Frank Ocean's  $\mu_j$  has a narrow interval; Lil Skies's  $\mu_j$  has a wide interval.
```

Posterior Prediction:

```
# Convert the simulation results to a data frame for easier manipulation
spotify_hierarchical_df <- as.data.frame(spotify_hierarchical)
```

Simulating Vampire Weekend's Posterior Predictive Model

```
# Extract relevant parameters and simulate predicted song popularities for Vampire Weekend
VW_chains <- spotify_hierarchical_df %>%
  rename(b = `b[(Intercept) artist:Vampire_Weekend]`) %>%
  select(`(Intercept)`, b, sigma) %>%
  mutate(
    # Calculate the mean popularity (mu_VW) for Vampire Weekend
    mu_VW = `(Intercept)` + b,
    # Simulate 20,000 draws for predicted song popularity
    y_VW = rnorm(20000, mean = mu_VW, sd = sigma)
  )
```

```
# Display the first few rows of the simulated predictions
head(VW_chains, 5)
```

```
##      (Intercept)          b      sigma      mu_VW      y_VW
## 1      48.53091 15.8766537 14.21260 64.40756 87.18861
## 2      48.85655  9.7863156 15.08145 58.64287 71.85871
## 3      49.26164 -0.7145565 14.03809 48.54708 39.44077
## 4      49.67779 12.0681560 14.20286 61.74594 64.06361
## 5      50.87859  8.5650047 13.59296 59.44360 41.32775
```

Summarize the posterior predictive distribution for Vampire Weekend

```
# Posterior summary for predicting the popularity of a new song
VW_chains %>%
  mean_qi(y_VW, .width = 0.80)
```

```
## # A tibble: 1 x 6
##   y_VW .lower .upper .width .point .interval
##   <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1  61.7  42.5  80.9    0.8 mean   qi
```

```
# Summarize the posterior for Vampire Weekend's mean popularity
artist_summary_scaled %>%
  filter(artist == "artist:Vampire_Weekend")
```

```
## # A tibble: 1 x 7
##   artist          mu_j .lower .upper .width .point .interval
##   <fct>          <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 artist:Vampire_Weekend 61.6  54.7  68.5    0.8 mean   qi
```

Insight: We can predict the mean popularity for Vampire Weekend with much higher precision than the popularity of any single new song due to the hierarchical model's shrinkage.

Posterior Prediction for a New Artist (Taylor Swift)

```
# Simulate the posterior predictive distribution for Taylor Swift
taylor_chains <- spotify_hierarchical_df %>%
  mutate(
    # Calculate the standard deviation for the artist-level intercept
    sigma_mu = sqrt(`Sigma[artist:(Intercept),(Intercept)]`),
    # Simulate the mean popularity for Taylor Swift
    mu_taylor = rnorm(20000, `(Intercept)`, sigma_mu),
    # Simulate predicted song popularities for Taylor Swift
    y_taylor = rnorm(20000, mu_taylor, sigma)
  )

# Posterior predictive summary for Taylor Swift
taylor_chains %>%
  mean_qi(y_taylor, .width = 0.80)
```

```
## # A tibble: 1 x 6
##   y_taylor .lower .upper .width .point .interval
##   <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1    52.5  25.7  79.2    0.8 mean   qi
```

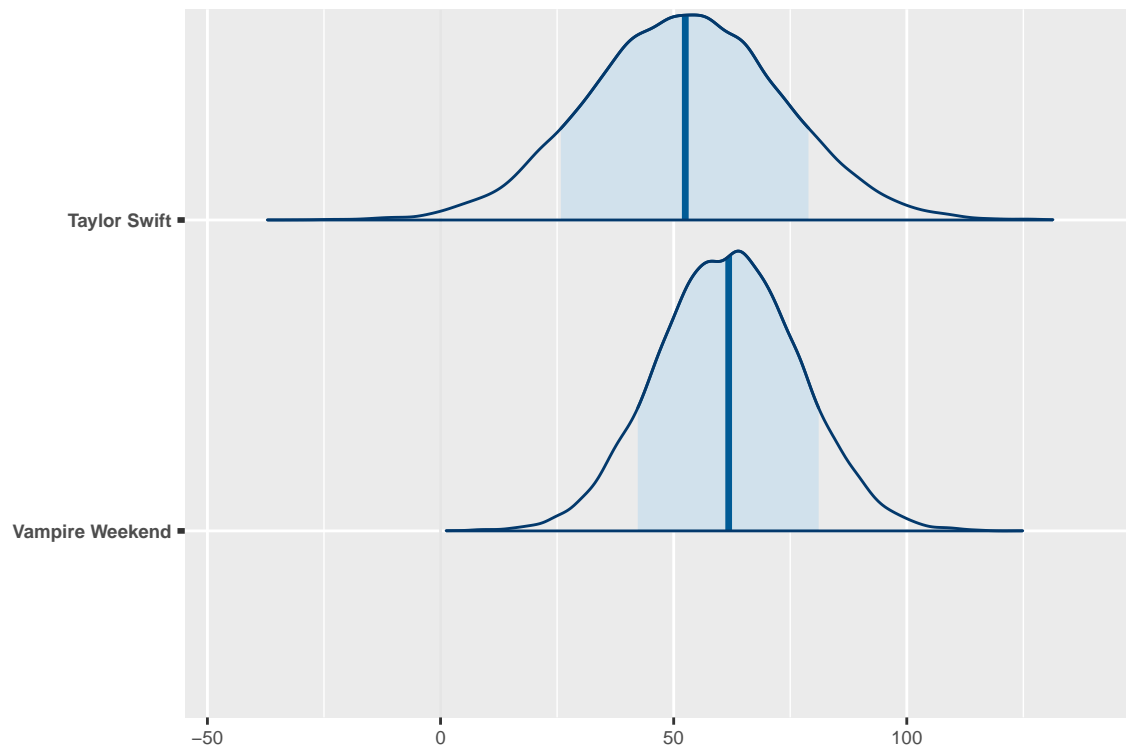
Summarizing Predictions for Multiple Artists

```
# Use a shortcut function to summarize predictions for both artists
prediction_shortcut <- posterior_predict(
  spotify_hierarchical,
  newdata = data.frame(artist = c("Vampire Weekend", "Taylor Swift"))
)

# Posterior predictive model plots for the two artists
mcmc_areas(prediction_shortcut, prob = 0.8) +
  ggplot2::scale_y_discrete(labels = c("Vampire Weekend", "Taylor Swift"))
```

Scale for y is already present.

Adding another scale for y, which will replace the existing scale.



Posterior Predictive Plots to Visualize Shrinkage

```
# Generate predictions for all artists based on their means
predictions_hierarchical <- posterior_predict(
  spotify_hierarchical,
  newdata = artist_means
)

# Plot posterior predictive intervals to demonstrate shrinkage
ppc_intervals(
  artist_means$mean_popularity,
  yrep = predictions_hierarchical,
  prob_outer = 0.80
) +
```



```

ggplot2::scale_x_continuous(
  labels = artist_means$artist,
  breaks = 1:nrow(artist_means)
) +
axis_text(angle = 90, hjust = 1) +
geom_hline(yintercept = 58.4, linetype = "dashed")

```

