Stat 479 PS03

Ruochong Fan

Dec 14 2021

Problem 1.

```
load("spotify_data.RData")
head(spotify_data)
```

```
# A tibble: 6 x 14
  track_name           track_artist genre popularity danceability energy loudness
  <chr>                <chr>        <chr>      <dbl>        <dbl>  <dbl>    <dbl>
1 busking              _tag         rap           62        0.622 0.0701   -13.0
2 En tête-à-tête - L~ -M-           rock          38        0.298 0.807     -5.77
3 One Girl / One Boy   !!!          pop           48        0.702 0.851     -5.75
4 Even When The Wate~ !!!           pop           52        0.709 0.831     -6.85
5 Ring Me Up - Sound~ !deladap      edm            5        0.773 0.902     -3.36
6 Music Has the Powe~ !deladap      edm           20        0.695 0.875     -4.11
# ... with 7 more variables: speechiness <dbl>, acousticness <dbl>,
#   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
#   duration <dbl>
```

Project prompt: What drives song popularity? First create a brief numeric summary for all columns (except artists and songs information).

```
summary(spotify_data[,4:14])
```

```
   popularity     danceability        energy            loudness
 Min.   : 0.00   Min.   :0.0000   Min.   :0.000175   Min.   :-46.448
 1st Qu.:25.00   1st Qu.:0.5640   1st Qu.:0.578000   1st Qu.: -8.242
 Median :43.50   Median :0.6710   Median :0.718000   Median : -6.228
 Mean   :41.13   Mean   :0.6552   Mean   :0.695517   Mean   : -6.779
 3rd Qu.:59.00   3rd Qu.:0.7610   3rd Qu.:0.839000   3rd Qu.: -4.699
 Max.   :98.00   Max.   :0.9830   Max.   :1.000000   Max.   :  1.275
  speechiness      acousticness     instrumentalness       liveness
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000000   Min.   :0.000
 1st Qu.:0.0411   1st Qu.:0.0153   1st Qu.:0.0000000   1st Qu.:0.093
 Median :0.0631   Median :0.0828   Median :0.0000182   Median :0.127
 Mean   :0.1084   Mean   :0.1804   Mean   :0.0901806   Mean   :0.190
 3rd Qu.:0.1340   3rd Qu.:0.2640   3rd Qu.:0.0058900   3rd Qu.:0.246
 Max.   :0.9180   Max.   :0.9940   Max.   :0.9940000   Max.   :0.996
    valence           tempo           duration
 Min.   :0.0000   Min.   :  0.00   Min.   :  4000
 1st Qu.:0.3260   1st Qu.: 99.99   1st Qu.:186682
 Median :0.5070   Median :121.99   Median :215053
 Mean   :0.5064   Mean   :120.97   Mean   :224705
 3rd Qu.:0.6890   3rd Qu.:134.00   3rd Qu.:252360
 Max.   :0.9905   Max.   :239.44   Max.   :517810
```

The data has 28553 rows and 14 columns. Here we can see that the response variable popularity has range [0, 100] with mean 41.13. Variables “danceability”, “energy”, “speechiness”, “acousticness”, “instrumentalness”, “liveness”, and “valence” are all numeric values with range [0, 1]. One of my top interests in songs is the variation between different music genres, where I am curious about which genre is currently top stream and therefore the popular. Here the 6 genres are listed on the table below. We can see that the 6 genres occur roughly with even numbers, so the level of genre is balanced when doing further modeling.

```
table(spotify_data$genre)
```

```
  edm latin   pop   r&b   rap  rock
 5335  4417  4836  4764  5273  3928
```

Here is the rough research question: Does the effect of variables danceability and valence on popularity vary among different song genres? In other words, the project aims to find possible association of danceability and valence with popularity and study the variation between different genres at the same time. The model used in this project is hierarchical linear regression model.

Reason: The variables popularity, danceability, valence are all continuous numeric values. Since the project aims to find association, it suitable to perform a linear regression model. By comparing the variance between each genre, I can set up a hierarchical model to analyze the data. Since both danceability and valence range [0, 1], I choose not to standardize the 2 predictors. In this case, the Spotify data is very suitable to answer the research question.

Problem 2.

Since the dataset has over 28,000 rows, it may be hard for RStan to run so many data. Therefore random sampling is performed. The project decides to pick the same number of data each genre for the sake of project adequacy. After trying several numbers, the project decides to randomly select 250 songs from the 6 genres: “rap” “rock” “pop” “edm” “latin” “r&b”. There are in total 1500 observed data. The goal for this project is to study possible variance of popularity under the given predictors between the 6 genres.

```
set.seed(479)
list_genre <- rep()
genres <- spotify_data$genre[!duplicated(spotify_data$genre)]
for (i in 1:length(genres)) {
  genre_filter <- spotify_data %>%
    filter(genre == genres[i])
  list_genre[[i]] <- sample_n(genre_filter, 250) # randomly select 250 data
}
```

Summary of the subsets

To give the selected data a closer look, the project creates a summary for each variable and compare it with the population data.

```
pop_sum <- rep()
dance_sum <- rep()
valence_sum <- rep()
for (i in 1:6) {
  pop_sum[[i]] <- summary(list_genre[[i]]$popularity)
  dance_sum[[i]] <- summary(list_genre[[i]]$danceability)
  valence_sum[[i]] <- summary(list_genre[[i]]$valence)
}
```

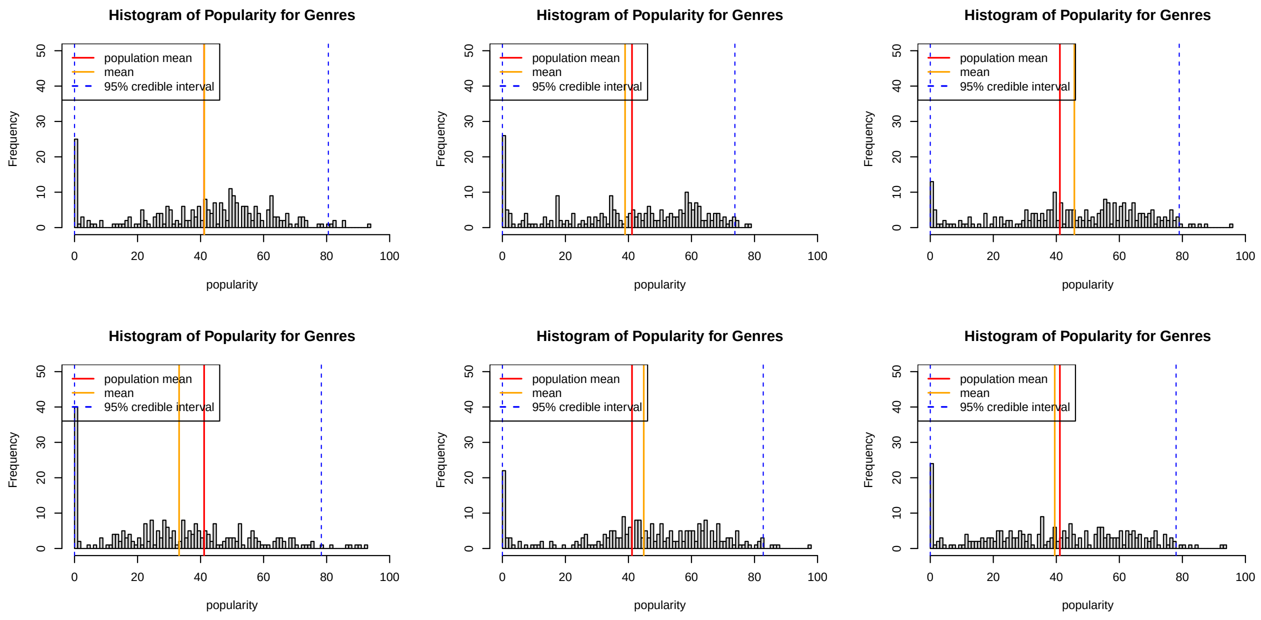
Popularity

```
pop_sum # numeric summary
```

```
[[1]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   28.00   45.00   41.18   56.00   94.00

[[2]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   19.00   42.00   38.93   59.00   79.00

[[3]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   33.00   48.00   45.72   62.75   96.00

[[4]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   16.00   32.00   33.19   49.00   93.00

[[5]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   34.00   47.00   44.85   63.00   98.00
```

```
[[6]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   0.00   22.00   40.00   39.49   58.75   94.00 
```

```
par(mfrow = c(2, 3))
for (i in 1:6) {
  hist(list_genre[[i]]$popularity, breaks = 100, xlab = "popularity",
       main = "Histogram of Popularity for Genres", xlim = c(0, 100), ylim = c(0, 50))
  abline(v = mean(spotify_data$popularity), col = "red", lwd = 1.5)
  abline(v = mean(list_genre[[i]]$popularity), col = "orange", lwd = 1.5) # mean
  abline(v = quantile(list_genre[[i]]$popularity, c(0.025, 0.975))[1],
         col = "blue", lty = "dashed") # 95% credible interval lower bound
  abline(v = quantile(list_genre[[i]]$popularity, c(0.025, 0.975))[2],
         col = "blue", lty = "dashed") # upper bound
  legend("topleft", legend = c("population mean", "mean", "95% credible interval"),
       lty = c(1, 1, 2), lwd = c(1.5, 1.5, 1.5), col = c('red', 'orange', 'blue'))
}
```



Here numbers 1-6 sequentially represent genres rap, rock, pop, EDM, Latin, and R&B. The population mean for popularity is 43.50. Here we can see that group 1, 2, 4, and 6 (rap, rock, EDM, and R&B) have mean popularity lower than the population mean. The mean for genre EDM is 33.19 and is significantly lower than other genres; the maximum popularity for rock is 79, also significantly lower than others. This may show that the EDM genre have some core fans but also have more unpopular songs since it has a high maximum popularity score but relatively low average (or may show that some EDM songs are really great but others are less popular); although rock has a higher popularity than EDM, not many people really like rock (may find it mediocre, fine to listen to but not great). Of course, the random sampling of 250 songs per genre is not very large and may cause bias and inaccuracy. Looking at the plot, the 2.5% quantile for all genres are located at 0. This may show that all genres have a lot of songs with popularity 0. In fact, 0 popularity is also the mode for all 6 genres of the selected data.

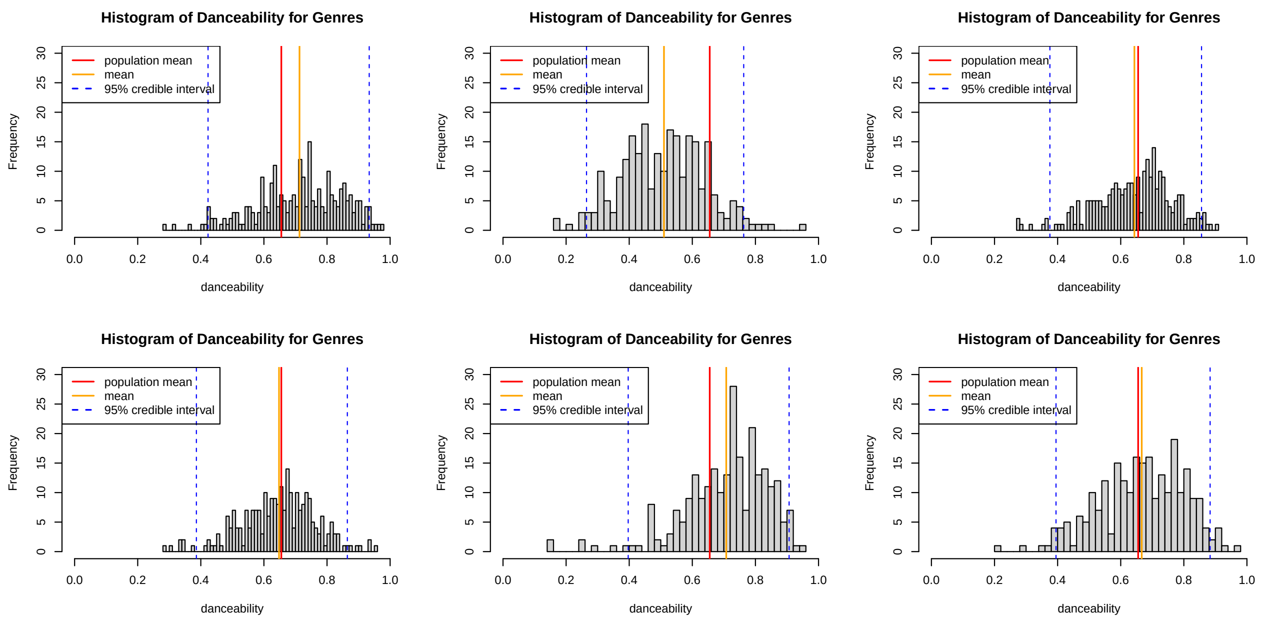
Danceability

```
dance_sum # numeric summary
```

```
[[1]]
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.2860  0.6270  0.7210  0.7128  0.8177  0.9720 

[[2]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.1660  0.4174  0.5135  0.5102  0.6040  0.9420 

[[3]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.2720  0.5687  0.6567  0.6434  0.7210  0.9080 

[[4]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.2850  0.5735  0.6555  0.6479  0.7258  0.9560 

[[5]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.1400  0.6252  0.7298  0.7075  0.8017  0.9440 

[[6]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.2190  0.5850  0.6770  0.6666  0.7688  0.9660 
```

```r
par(mfrow = c(2, 3))
for (i in 1:6) {
  hist(list_genre[[i]]$danceability, breaks = 50, xlab = "danceability",
       main = "Histogram of Danceability for Genres", xlim = c(0, 1), ylim = c(0, 30))
  abline(v = mean(spotify_data$danceability), col = "red", lwd = 1.5) # population mean
  abline(v = mean(list_genre[[i]]$danceability), col = "orange", lwd = 1.5) # mean
  abline(v = quantile(list_genre[[i]]$danceability, c(0.025, 0.975))[1],
         col = "blue", lty = "dashed") # 95% credible interval lower bound
  abline(v = quantile(list_genre[[i]]$danceability, c(0.025, 0.975))[2],
         col = "blue", lty = "dashed") # upper bound
  legend("topleft", legend = c("population mean", "mean", "95% credible interval"),
       lty = c(1, 1, 2), lwd = c(1.5, 1.5, 1.5), col = c('red', 'orange', 'blue'))
}
```



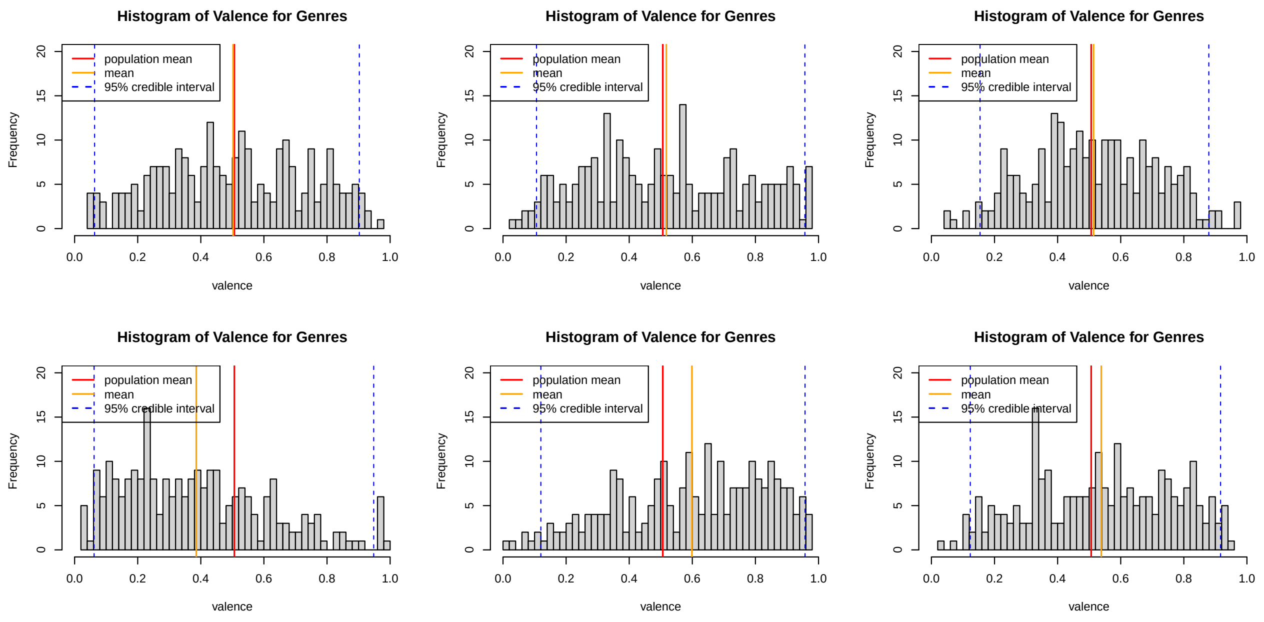
The variable danceability is an interesting predictor. Briefly thinking, not all songs are suitable for dancing but can still be very popular; on the other hand, attractive dancing videos can gain a lot of popularity for a song. While looking at the data summary, danceability has population mean 0.655. Genres 2 (rock) has mean significantly lower than the population mean. We can also see that genre rap and Latin have generally higher mean danceability. If the predictor danceability has a large and positive effect on the popularity score, than we would expect songs from genre rap and Latin to be more popular than others.

Valence

```
valence_sum
```

```
[[1]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0410  0.3220  0.5055  0.5023  0.6825  0.9636 

[[2]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0383  0.3212  0.5035  0.5176  0.7255  0.9700 

[[3]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0438  0.3817  0.5030  0.5138  0.6689  0.9720 

[[4]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0342  0.2045  0.3540  0.3856  0.5377  0.9830 

[[5]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
0.00001 0.41700 0.62600 0.59885 0.79600 0.96900 

[[6]]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0395  0.3500  0.5445  0.5384  0.7340  0.9600 
```

```
par(mfrow = c(2, 3))
for (i in 1:6) {
  hist(list_genre[[i]]$valence, breaks = 50, xlab = "valence",
       main = "Histogram of Valence for Genres", xlim = c(0, 1), ylim = c(0, 20))
  abline(v = mean(spotify_data$valence), col = "red", lwd = 1.5) # population mean
  abline(v = mean(list_genre[[i]]$valence), col = "orange", lwd = 1.5) # mean
  abline(v = quantile(list_genre[[i]]$valence, c(0.025, 0.975))[1],
         col = "blue", lty = "dashed") # 95% credible interval lower bound
  abline(v = quantile(list_genre[[i]]$valence, c(0.025, 0.975))[2],
         col = "blue", lty = "dashed") # upper bound
  legend("topleft", legend = c("population mean", "mean", "95% credible interval"),
       lty = c(1, 1, 2), lwd = c(1.5, 1.5, 1.5), col = c('red', 'orange', 'blue'))
}
```

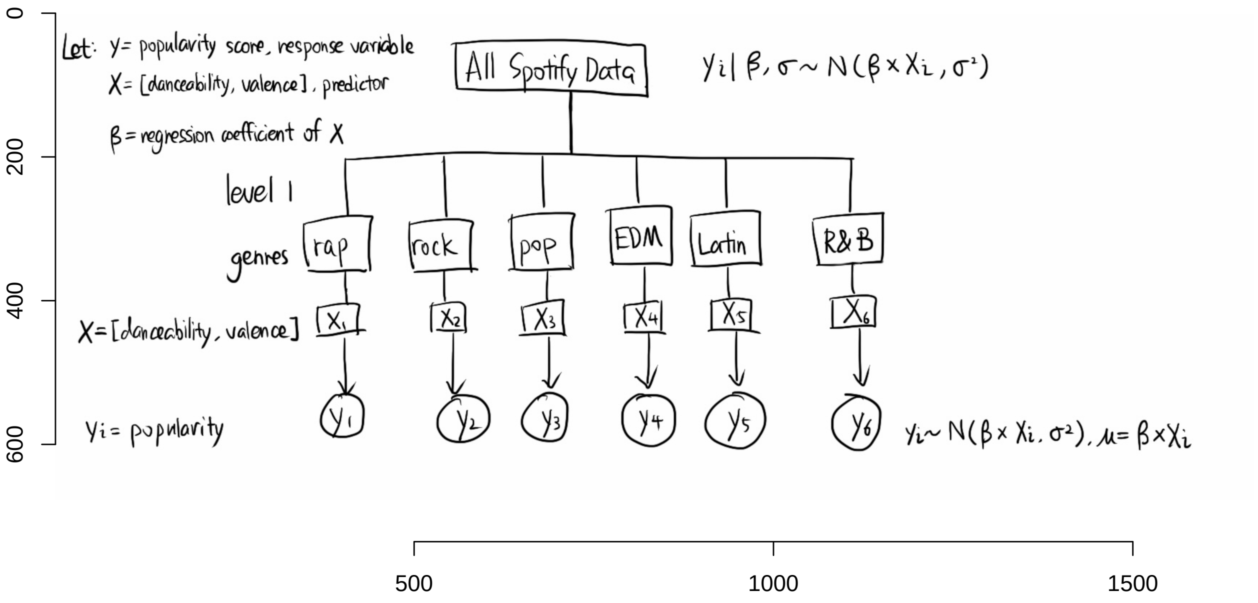


The population mean for variable valence is 0.5064. We can see that genre EDM has mean valence much lower than the population mean and genre Latin has mean valence much higher than population mean. Here all valence data points are located around the population mean but have different skew patterns. This may show that valence may vary a lot among different genres. It remains unclear at this stage whether valence is positively or negatively associated with popularity. If valence is positive associated with popularity, then genre Latin may have a higher popularity score; on the other hand, genre EDM may generally have a higher popularity if valence is negatively associated with popularity. Based on current information, the project would expect predictor danceability to have a stronger effect than valence. The project would also expect that valence to show significant variation between genres.

Problem 3.

Draw a Hierarchical Diagram

```r
hier_diagram <- load.image("hier_diagram.png")
plot(hier_diagram)
```


Let: y= popularity score, response variable
X= [danceability, valence], predictor
β= regression coefficient of X
All Spotify Data
$y_i | \beta, \sigma \sim N(\beta \times X_i, \sigma^2)$
level 1
genres
rap
rock
pop
EDM
Latin
R&B
X=[danceability, valence]
X_1
X_2
X_3
X_4
X_5
X_6
y_i= popularity
y_1
y_2
y_3
y_4
y_5
y_6
$y_i \sim N(\beta \times X_i, \sigma^2), \mu = \beta \times X_i$
0
200
400
600
500
1000
1500

The Hierarchical model has 2 levels. The Spotify data is specified into level 1 of 6 genres. Using the hierarchical linear regression model, this project aims to analyze the effect of danceability and valence on popularity and how the effect varies between genres.

Mathematical Model

As partially shown in the diagram plot, the model this project uses is linear regression. The linear model can be roughly presented as $y_i|\beta, \sigma \sim N(\beta \times X_{ij}, \sigma^2)$. Since the minimum popularity is 0, the project does not find it necessary to set an α (representing the baseline) for the linear model. The response variable y is a matrix with dim(N, 1) where N is the number of observed data all genres. The matrix X has 2 predictors: danceability and valence with dim(N, 2). The regression coefficient β has dim(2, 1). Since the project chooses a hierarchical model with 1 nested structure and 6 groups, $X_{ij}, i \in [1, 6], j \in [1, 2]$ specifies the predictors for each group. Here the index i represents the i^{th} group (genre) while $j = 1$ for predictor danceability and $j = 2$ for predictor valence. The term σ is the standard deviation of the response variable popularity. Again since both chosen predictors ranges from 0 to 1, the project chooses not to perform standardization.

Model Description.

The data can be generated using the mean and standard deviation. For the response variable y, std $y_i = \frac{y_i - \mu_{y_i}}{\sigma_{y_i}}$ and the same for X_{ij}. Since the project chooses not to perform standardization, there is no need to input the mean and standard deviation of variables.

Problem 4.

Fitting a prior: I do not have a lot of prior music knowledge so the project references population data (here is the complete Spotify dataset) for prior choices. According to the data summaries, the project believes that there is variation between the effect of danceability and the effect valence on popularity. Therefore the project decides to set different priors for the 2 predictors. Generally, the project expects the prior regression coefficients (β_1, β_2 in this case) to follow a normal distribution. The prior model looks like

$$y_i(\text{popularity}) = \text{danceability} \times \beta_1 + \text{valence} \times \beta_2$$

To specify the mean and variance for the 2 prior regression coefficients, we can look at the change in response when increasing the predictor by 1 standard deviation or by 1 unit. The project gives all 6 genres the same prior for β_1 and β_2 since the project does not assume variation among genres before fitting the model. Since the model has 2 predictors, the project decides to fix one predictor at mean and look at the other. After some rounds of observations, the project sets the mean for β_1 (prior regression coefficient for danceability) at about 75 and set mean β_2 at about 6. For the variance, the project expects the term σ^2 to be relative small for the sake of model accuracy. From past experience, the project observes that larger variance in prior coefficients tend to give larger credible intervals for the posterior – which means that the more vague results. The final choice of priors are $\beta_1 \sim \mathbb{N}(75, 0.5)$ and $\beta_2 \sim \mathbb{N}(6.2, 0.6)$. Here the project references Stan user guide record[1] for some Stan codes and logic for the hierarchical structure.

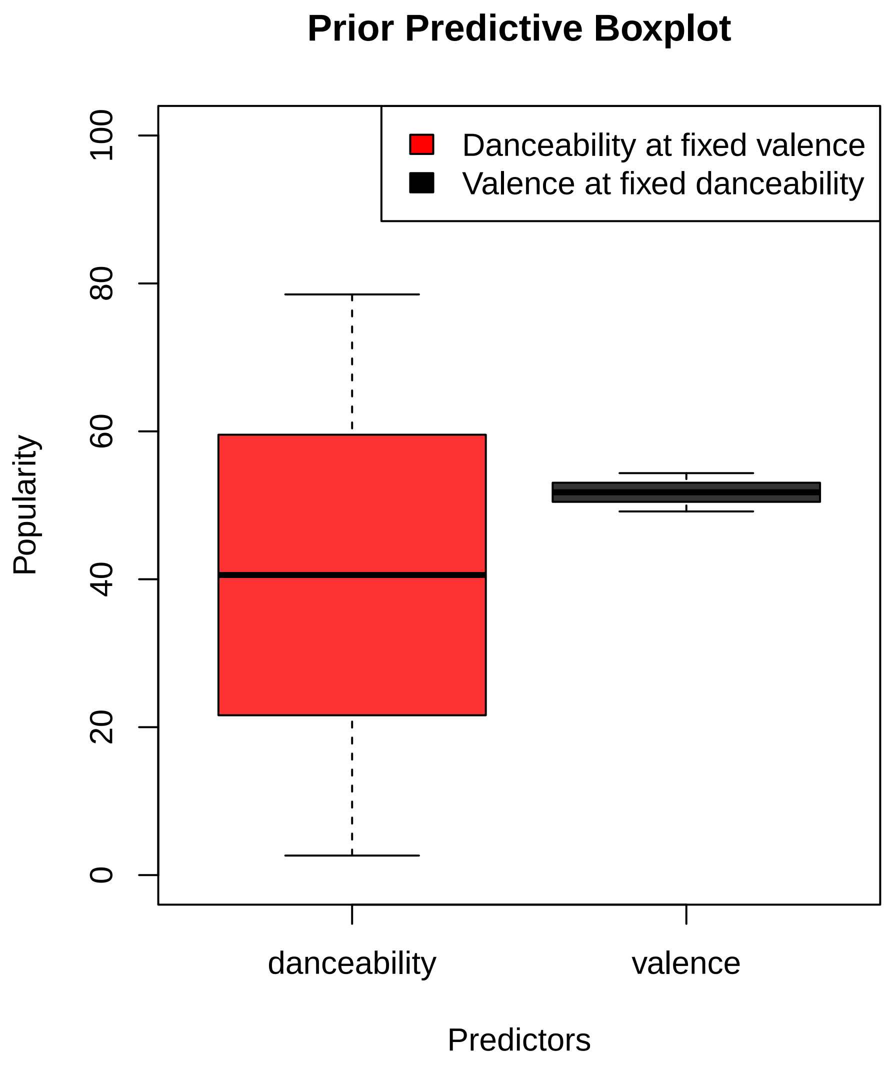
Another prior that needs to be specified is response's standard deviation. From the previous section, we can see that the model can also be expressed like $y_i \sim N((\mu = X_i \times \beta), \sigma^2)$. Since the RStan codes can compute the mean, the project still needs to specify σ^2. Here the project expects that there is relatively large variance on the response since it ranges from 0 to 100. After some observations, the prior is $\sigma^2 \sim N(23, 1)$.

Prior Predictive Popularity

To create a 2D plot with 2 predictors, the project decides to fix one predictor at certain values and put the other on x-axis. Here the project decides to fix a certain predictor at its mean (neglect genre differences at this point since we have not fit the model to find genres variations at this stage).

```r
set.seed(479)
beta_1_prior <- rnorm(1, mean = 75, sd = 0.5) # randomly draw 1 sample
beta_2_prior <- rnorm(1, mean = 6.2, sd = 0.6)
merged_data <- rbind(list_genre[[1]], list_genre[[2]], list_genre[[3]],
                     list_genre[[4]], list_genre[[5]], list_genre[[6]])
prior_pred_grid <- seq(0, 1, by = 0.01)

dance_prior <- beta_1_prior * prior_pred_grid + beta_2_prior * mean(merged_data$valence)
valence_prior <- beta_1_prior * mean(merged_data$danceability) + beta_2_prior * prior_pred_grid
boxplot(dance_prior, valence_prior, col = alpha(c("red", "black"), 0.8),
        names = c("danceability", "valence"), main = "Prior Predictive Boxplot",
        ylim = c(0, 100), xlab = "Predictors", ylab = "Popularity")
legend("topright", fill = c("red", "black"),
       legend = c("Danceability at fixed valence", "Valence at fixed danceability"), horiz = F)
```

[1] https://mc-stan.org/docs/2_19/stan-users-guide/multivariate-hierarchical-priors-section.html



From the plots, we can see that predictor valence tend to have higher predicted valence when danceability is fixed at its mean. The box for valence is also much denser than danceability's box. This may show that the prior prediction has generally small variance and high certainty. Another thing to notice is that the left box (danceability) has much higher maximum and minimum prediction, meaning that it covers a broader data points.

Problem 5.

First set up the dimension of matrices:

```r
N <- 1500 # sample size
J <- 6 # number of genres
id <- rep(1:J, each = 250) # genre indices
K <- 2 # number of predictors
```

Create a list:

```r
# predictor
danceability <- rep()
valence <- rep()
for (i in 1:6) {
  danceability[[i]] <- list_genre[[i]] %>% pull(danceability)
  valence[[i]] <- list_genre[[i]] %>% pull(valence)
}
dance_list <- c(danceability[[1]], danceability[[2]], danceability[[3]],
                danceability[[4]], danceability[[5]], danceability[[6]])
valence_list <- c(valence[[1]], valence[[2]], valence[[3]],
                  valence[[4]], valence[[5]], valence[[6]])
X <- cbind(dance_list, valence_list) # the predictor matrix
# response
popularity <- rep()
for (i in 1:6) {
  popularity[[i]] <- list_genre[[i]] %>% pull(popularity)
}
y <- c(popularity[[1]], popularity[[2]], popularity[[3]],
       popularity[[4]], popularity[[5]], popularity[[6]])
# prediction
dance_grid <- seq(0, 1, by = 0.01)
valence_grid <- seq(0, 1, by = 0.01)
n_grid <- length(dance_grid)
```

The project fits corresponding X and y data by genre order of rap, rock, pop, EDM, Latin, and R&B. Now we can run the Stan model and check for divergence.

```r
data_list = list(N = N,
                 J = J,
                 K = K,
                 id = id,
                 X = X,
                 y = y,
                 n_grid = n_grid,
                 dance_grid = dance_grid,
                 valence_grid = valence_grid)
hier_stan <- stan_model(file = "hier_lm.stan")
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Li
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeaders/include
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/E
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/E
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Core/util/Mac
```

```
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Core/util/Ma
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeaders/includ
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/Core:96:10: fata
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1
```

```r
hier_fit <- sampling(object = hier_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'hier_lm' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.00049 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 4.9 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 6.38234 seconds (Warm-up)
Chain 1:                3.77788 seconds (Sampling)
Chain 1:                10.1602 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'hier_lm' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000344 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 3.44 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
```

```
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 5.33606 seconds (Warm-up)
Chain 2:                3.77561 seconds (Sampling)
Chain 2:                9.11167 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'hier_lm' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000324 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 3.24 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 6.20064 seconds (Warm-up)
Chain 3:                3.77157 seconds (Sampling)
Chain 3:                9.97221 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'hier_lm' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000334 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 3.34 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4: 
Chain 4:  Elapsed Time: 5.67686 seconds (Warm-up)
Chain 4:                3.75064 seconds (Sampling)
Chain 4:                9.42749 seconds (Total)
Chain 4: 
```

The output Rhat values are to long so the project decides to put it in Appendix for file coherency. The Rhat values seem to be close to 1 and none exceed 1.1 – showing a convergence trend. Here the project uses the default sampling setting of 4 chains and 2000 iterations (including 1000 warm-ups). The hierarchical structure is achieved in the Stan model by indexing different song genres.

Problem 6.

Summarize the findings. First extract the parameters:

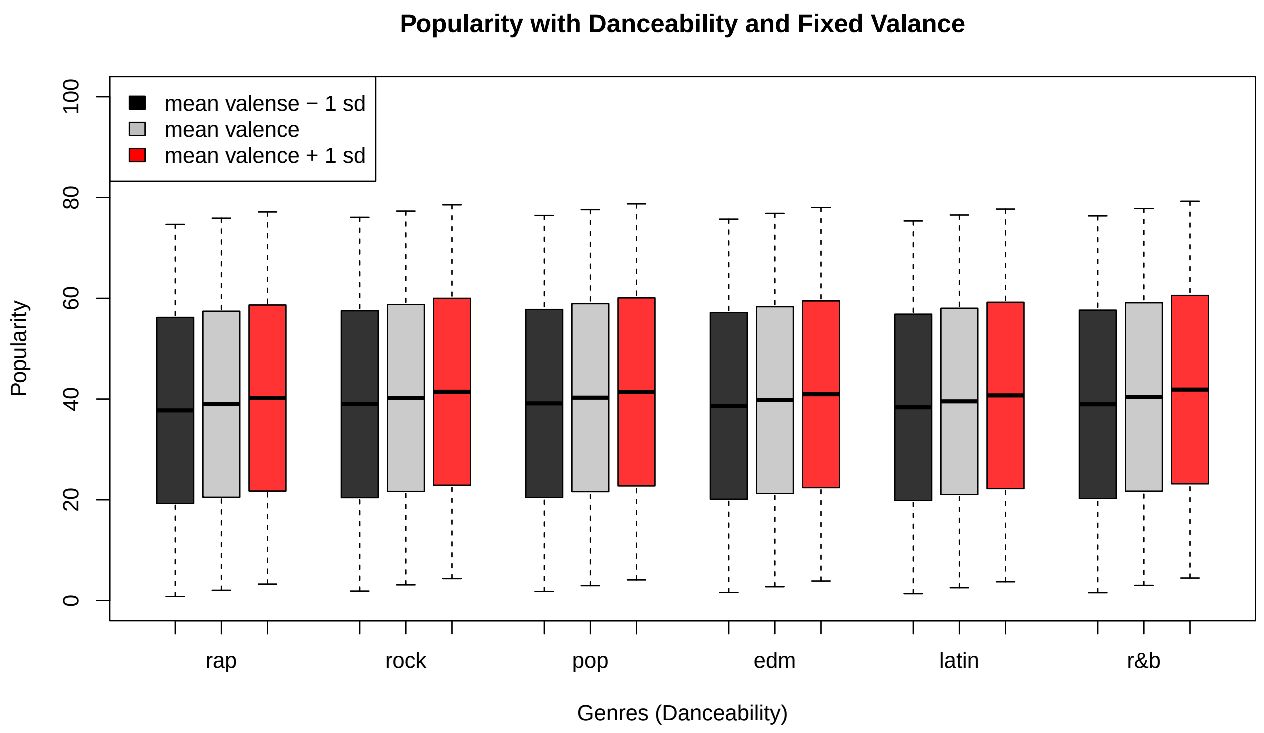
```
post_matrix <- rstan::extract(hier_fit, pars = "post_matrix")[["post_matrix"]]
beta <- rstan::extract(hier_fit, pars = "beta")[["beta"]]
c(mean(beta[,,1]), mean(beta[,,2]))
```

```
[1] 74.270252  5.365844
```

Perform comparative box plots

Danceability on x-axis

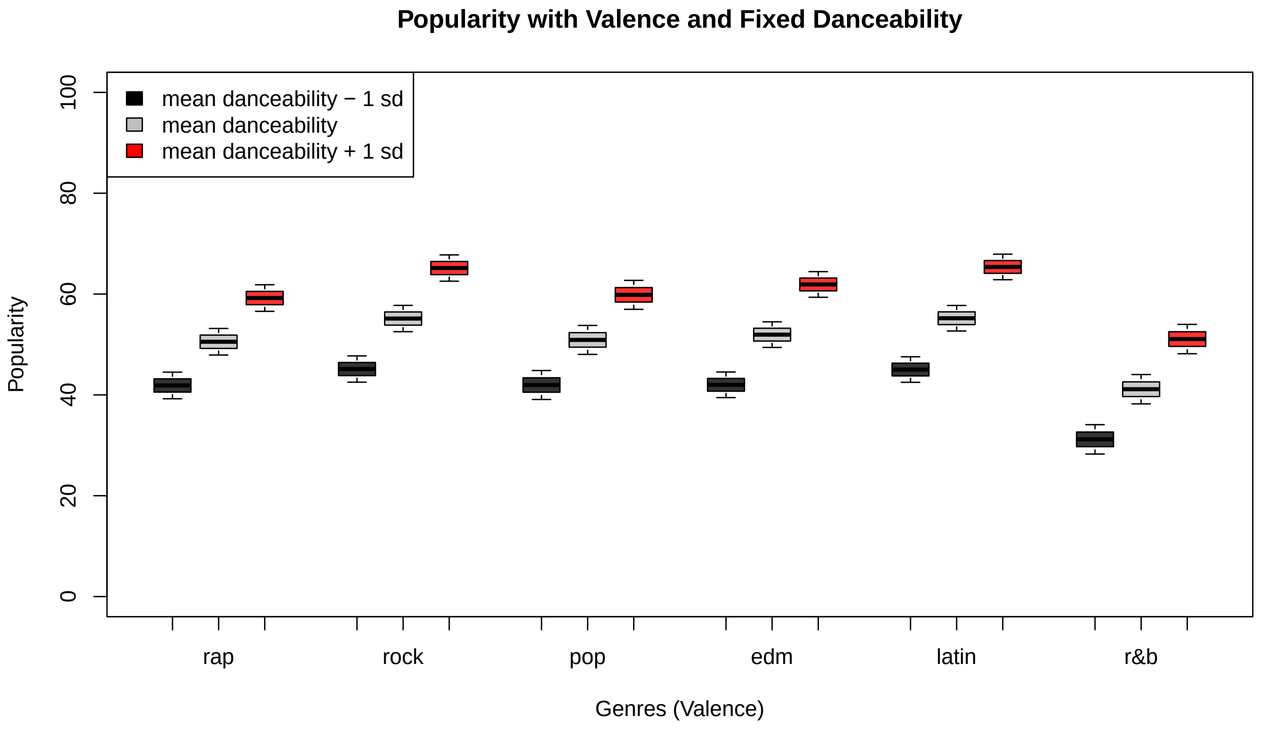
```
# set valance at mean
idx <- c(1, 251, 501, 751, 1001, 1251, 1500)
# mean
dance_mean <- rep()
for (i in 1:6) {
  dance_mean[[i]] <- mean(beta[,i,1]) * dance_grid +
    mean(beta[,i,2]) * mean(X[idx[i]:idx[i+1], 2])
}
# mean - 1sd
dance_minus_sd <- rep()
for (i in 1:6) {
  dance_minus_sd[[i]] <- mean(beta[,i,1]) * dance_grid +
    mean(beta[,i,2]) * (mean(X[idx[i]:idx[i+1], 2]) - sd(X[idx[i]:idx[i+1], 2]))
}
# mean + 1sd
dance_plus_sd <- rep()
for (i in 1:6) {
  dance_plus_sd[[i]] <- mean(beta[,i,1]) * dance_grid +
    mean(beta[,i,2]) * (mean(X[idx[i]:idx[i+1], 2]) + sd(X[idx[i]:idx[i+1], 2]))
}
# create a data frame
df_dance <- data.frame(x = c(c(dance_minus_sd[[1]], dance_mean[[1]], dance_plus_sd[[1]]),
                             c(dance_minus_sd[[2]], dance_mean[[2]], dance_plus_sd[[2]]),
                             c(dance_minus_sd[[3]], dance_mean[[3]], dance_plus_sd[[3]]),
                             c(dance_minus_sd[[4]], dance_mean[[4]], dance_plus_sd[[4]]),
                             c(dance_minus_sd[[5]], dance_mean[[5]], dance_plus_sd[[5]]),
                             c(dance_minus_sd[[6]], dance_mean[[6]], dance_plus_sd[[6]])),
                       y = rep(genres, each = 303),
                       z = rep(rep(1:3, each = 101), 6),
                       stringsAsFactors = FALSE)
# boxplot
cols <- c("black", "grey", "red")
boxplot(x ~ z + y, data = df_dance,
        at = c(1:3, 5:7, 9:11, 13:15, 17:19, 21:23), col = alpha(cols, 0.8),
        main = "Popularity with Danceability and Fixed Valance",
        xlab = "Genres (Danceability)", ylab = "Popularity",
        names = c("", "rap", "", "", "rock", "", "", "pop", "",
                  "", "edm", "", "", "latin", "", "", "r&b", ""),
        xaxs = FALSE, ylim = c(0, 100))
legend("topleft", fill = cols,
       legend = c("mean valense - 1 sd", "mean valence", "mean valence + 1 sd"),
        horiz = F)
```



Here we can see that the maximum predicted popularity scores reach about 80 but the lowest popularity reaches negative for genre EDM and Latin. From the grouped boxplots we can see that there may exist variation on popularity among genres but the variation is not significant. The project is acknowledged of the limited selected samples (in total 1500 observed data) therefore the variation can also be caused by selection bias. Apart from the bias, R&B generally has a higher popularity and Latin and EDM seem to have a lower popularity. The predictor danceability's effect on popularity seem to be significant.

Valence on x-axis

```
# mean
val_mean <- rep()
for (i in 1:6) {
  val_mean[[i]] <- mean(beta[,i,1]) * mean(X[idx[i]:idx[i+1], 1]) +
    mean(beta[,i,2]) * valence_grid
}
# mean - 1sd
val_minus_sd <- rep()
for (i in 1:6) {
  val_minus_sd[[i]] <- mean(beta[,i,1]) *
    (mean(X[idx[i]:idx[i+1], 1]) - sd(X[idx[i]:idx[i+1], 1])) +
    mean(beta[,i,2]) * valence_grid
}
# mean + 1sd
val_plus_sd <- rep()
for (i in 1:6) {
  val_plus_sd[[i]] <- mean(beta[,i,1]) *
    (mean(X[idx[i]:idx[i+1], 1]) + sd(X[idx[i]:idx[i+1], 1])) +
    mean(beta[,i,2]) * valence_grid
}
# create a data frame
df_valence <- data.frame(x = c(c(val_minus_sd[[1]], val_mean[[1]], val_plus_sd[[1]]),
                               c(val_minus_sd[[2]], val_mean[[2]], val_plus_sd[[2]]),
                               c(val_minus_sd[[3]], val_mean[[3]], val_plus_sd[[3]]),
                               c(val_minus_sd[[4]], val_mean[[4]], val_plus_sd[[4]]),
                               c(val_minus_sd[[5]], val_mean[[5]], val_plus_sd[[5]]),
                               c(val_minus_sd[[6]], val_mean[[6]], val_plus_sd[[6]])),
                         y = rep(genres, each = 303),
                         z = rep(rep(1:3, each = 101), 6),
                         stringsAsFactors = FALSE)
# boxplot
cols <- c("black", "grey", "red")
boxplot(x ~ z + y, data = df_valence,
        at = c(1:3, 5:7, 9:11, 13:15, 17:19, 21:23), col = alpha(cols, 0.8),
        main = "Popularity with Valence and Fixed Danceability",
        xlab = "Genres (Valence)", ylab = "Popularity",
        names = c("", "rap", "", "", "rock", "", "", "pop", "",
                  "", "edm", "", "", "latin", "", "", "r&b", ""),
        xaxs = FALSE, ylim = c(0, 100))
        #cex.lab = 2, cex.axis = 2, cex.main = 2, cex.sub = 2
legend("topleft", fill = cols,
       legend = c("mean danceability - 1 sd", "mean danceability",
                  "mean danceability + 1 sd"), horiz = F)
```



The effect of valence on popularity does not seem very significant as shown by the regression coefficient and by the plot. Here we can see that the posterior predictive popularity scores are generally located around 30 to 70. Rock songs and Latin songs have the highest popularity as predicted and R&B songs seem to have the lowest popularity. Here notices that the selected pop songs tend to show a really skinny and dense posterior prediction, where the box's size is skewed and all upper bound, mean, lower bound are focused between 40 and 60 popularity score. It may be hard to predict popularity close to either 0 or 1.

Looking at the 2 plots together, we can primarily conclude that there is clear popularity variation between genres by the effect of danceability and valence. Looking at the valence axis, there is generally a stronger variance than the danceability axis Generally speaking, Latin songs seem to have a higher popularity based on the two predictors and rock songs seem to be the second highest.

Problem 7.

To check model adequacy, we can calculate the MSE and MSM.

```r
post_pred <- rep()
for (i in 1:6) {
  post_pred[[i]] <- mean(beta[,i,1]) * X[idx[i]:idx[i+1], 1] +
    mean(beta[,i,2]) * X[idx[i]:idx[i+1], 2]
}
# merge the list
merged_post_grid <- c(post_pred[[1]], post_pred[[2]], post_pred[[3]],
  post_pred[[4]], post_pred[[5]], post_pred[[6]])
(MSE <- mean((y - merged_post_grid)^2))
```

```
[1] 747.1992
```

```r
# calculate the MSM
(MSM <- sum((merged_post_grid - mean(y))^2))
```

```
[1] 350007.8
```

As we can see, the MSE is relatively large by comparing it to the MSM. The model may not be a very adequate summary of the data. From the model, we can see that predictor “danceability” has a larger effect on popularity than “valence”. Valence, however, has a higher mean popularity prediction.

Modifications: To improve the model, I can either add more predictors or consider more complex hierarchical structures. Here the hierarchical structure only contains 2 levels with one nested group: genres and songs. Some improvements can be introducing more levels like songs nested within artists within genres; the project can also consider geographical and language variables to improve the model. Also, the project only includes 2 predictors, which may not be a good fit of all data. Introducing more variables or compute feature selection may be one way to improve.

Appendix

```r
summary(hier_fit)[[1]][,"Rhat"]
```

```
        beta[1,1]         beta[1,2]         beta[2,1]         beta[2,2] 
        0.9998383         0.9993055         0.9995266         0.9995259 
        beta[3,1]         beta[3,2]         beta[4,1]         beta[4,2] 
        0.9990642         0.9993369         0.9993126         0.9994592 
        beta[5,1]         beta[5,2]         beta[6,1]         beta[6,2] 
        0.9992998         0.9992111         0.9994727         0.9996081 
            sigma  post_matrix[1,1]  post_matrix[1,2]  post_matrix[1,3] 
        0.9993042         0.9993055         0.9995259         0.9993369 
 post_matrix[1,4]  post_matrix[1,5]  post_matrix[1,6]  post_matrix[2,1] 
        0.9994592         0.9992111         0.9996081         0.9993059 
 post_matrix[2,2]  post_matrix[2,3]  post_matrix[2,4]  post_matrix[2,5] 
        0.9995161         0.9993373         0.9994578         0.9992141 
 post_matrix[2,6]  post_matrix[3,1]  post_matrix[3,2]  post_matrix[3,3] 
        0.9996132         0.9993066         0.9995064         0.9993375 
 post_matrix[3,4]  post_matrix[3,5]  post_matrix[3,6]  post_matrix[4,1] 
        0.9994562         0.9992172         0.9996181         0.9993076 
 post_matrix[4,2]  post_matrix[4,3]  post_matrix[4,4]  post_matrix[4,5] 
        0.9994966         0.9993376         0.9994546         0.9992204 
 post_matrix[4,6]  post_matrix[5,1]  post_matrix[5,2]  post_matrix[5,3] 
        0.9996230         0.9993088         0.9994868         0.9993375 
 post_matrix[5,4]  post_matrix[5,5]  post_matrix[5,6]  post_matrix[6,1] 
        0.9994529         0.9992235         0.9996279         0.9993103 
 post_matrix[6,2]  post_matrix[6,3]  post_matrix[6,4]  post_matrix[6,5] 
        0.9994771         0.9993373         0.9994511         0.9992267 
 post_matrix[6,6]  post_matrix[7,1]  post_matrix[7,2]  post_matrix[7,3] 
        0.9996326         0.9993120         0.9994674         0.9993369 
 post_matrix[7,4]  post_matrix[7,5]  post_matrix[7,6]  post_matrix[8,1] 
        0.9994493         0.9992300         0.9996373         0.9993140 
 post_matrix[8,2]  post_matrix[8,3]  post_matrix[8,4]  post_matrix[8,5] 
        0.9994577         0.9993364         0.9994474         0.9992332 
 post_matrix[8,6]  post_matrix[9,1]  post_matrix[9,2]  post_matrix[9,3] 
        0.9996418         0.9993163         0.9994481         0.9993357 
 post_matrix[9,4]  post_matrix[9,5]  post_matrix[9,6] post_matrix[10,1] 
        0.9994455         0.9992365         0.9996463         0.9993187 
post_matrix[10,2] post_matrix[10,3] post_matrix[10,4] post_matrix[10,5] 
        0.9994385         0.9993348         0.9994435         0.9992398 
post_matrix[10,6] post_matrix[11,1] post_matrix[11,2] post_matrix[11,3] 
        0.9996506         0.9993215         0.9994291         0.9993338 
post_matrix[11,4] post_matrix[11,5] post_matrix[11,6] post_matrix[12,1] 
        0.9994414         0.9992431         0.9996548         0.9993244 
post_matrix[12,2] post_matrix[12,3] post_matrix[12,4] post_matrix[12,5] 
        0.9994197         0.9993326         0.9994394         0.9992464 
post_matrix[12,6] post_matrix[13,1] post_matrix[13,2] post_matrix[13,3] 
        0.9996589         0.9993276         0.9994105         0.9993313 
post_matrix[13,4] post_matrix[13,5] post_matrix[13,6] post_matrix[14,1] 
        0.9994373         0.9992497         0.9996629         0.9993310 
post_matrix[14,2] post_matrix[14,3] post_matrix[14,4] post_matrix[14,5] 
        0.9994014         0.9993299         0.9994351         0.9992530 
post_matrix[14,6] post_matrix[15,1] post_matrix[15,2] post_matrix[15,3] 
        0.9996667         0.9993346         0.9993925         0.9993283 
```

```
post_matrix[15,4] post_matrix[15,5] post_matrix[15,6] post_matrix[16,1] 
        0.9994329         0.9992562         0.9996704         0.9993384 
post_matrix[16,2] post_matrix[16,3] post_matrix[16,4] post_matrix[16,5] 
        0.9993837         0.9993266         0.9994307         0.9992595 
post_matrix[16,6] post_matrix[17,1] post_matrix[17,2] post_matrix[17,3] 
        0.9996739         0.9993424         0.9993751         0.9993247 
post_matrix[17,4] post_matrix[17,5] post_matrix[17,6] post_matrix[18,1] 
        0.9994285         0.9992628         0.9996773         0.9993465 
post_matrix[18,2] post_matrix[18,3] post_matrix[18,4] post_matrix[18,5] 
        0.9993667         0.9993227         0.9994262         0.9992660 
post_matrix[18,6] post_matrix[19,1] post_matrix[19,2] post_matrix[19,3] 
        0.9996806         0.9993509         0.9993585         0.9993207 
post_matrix[19,4] post_matrix[19,5] post_matrix[19,6] post_matrix[20,1] 
        0.9994240         0.9992692         0.9996836         0.9993553 
post_matrix[20,2] post_matrix[20,3] post_matrix[20,4] post_matrix[20,5] 
        0.9993504         0.9993185         0.9994217         0.9992723 
post_matrix[20,6] post_matrix[21,1] post_matrix[21,2] post_matrix[21,3] 
        0.9996866         0.9993600         0.9993426         0.9993162 
post_matrix[21,4] post_matrix[21,5] post_matrix[21,6] post_matrix[22,1] 
        0.9994194         0.9992754         0.9996893         0.9993648 
post_matrix[22,2] post_matrix[22,3] post_matrix[22,4] post_matrix[22,5] 
        0.9993350         0.9993138         0.9994171         0.9992785 
post_matrix[22,6] post_matrix[23,1] post_matrix[23,2] post_matrix[23,3] 
        0.9996920         0.9993697         0.9993276         0.9993113 
post_matrix[23,4] post_matrix[23,5] post_matrix[23,6] post_matrix[24,1] 
        0.9994148         0.9992816         0.9996944         0.9993747 
post_matrix[24,2] post_matrix[24,3] post_matrix[24,4] post_matrix[24,5] 
        0.9993204         0.9993087         0.9994125         0.9992845 
post_matrix[24,6] post_matrix[25,1] post_matrix[25,2] post_matrix[25,3] 
        0.9996967         0.9993798         0.9993135         0.9993060 
post_matrix[25,4] post_matrix[25,5] post_matrix[25,6] post_matrix[26,1] 
        0.9994102         0.9992875         0.9996989         0.9993850 
post_matrix[26,2] post_matrix[26,3] post_matrix[26,4] post_matrix[26,5] 
        0.9993068         0.9993033         0.9994079         0.9992904 
post_matrix[26,6] post_matrix[27,1] post_matrix[27,2] post_matrix[27,3] 
        0.9997009         0.9993904         0.9993004         0.9993005 
post_matrix[27,4] post_matrix[27,5] post_matrix[27,6] post_matrix[28,1] 
        0.9994057         0.9992932         0.9997027         0.9993958 
post_matrix[28,2] post_matrix[28,3] post_matrix[28,4] post_matrix[28,5] 
        0.9992942         0.9992977         0.9994034         0.9992960 
post_matrix[28,6] post_matrix[29,1] post_matrix[29,2] post_matrix[29,3] 
        0.9997044         0.9994012         0.9992882         0.9992948 
post_matrix[29,4] post_matrix[29,5] post_matrix[29,6] post_matrix[30,1] 
        0.9994011         0.9992987         0.9997060         0.9994067 
post_matrix[30,2] post_matrix[30,3] post_matrix[30,4] post_matrix[30,5] 
        0.9992824         0.9992918         0.9993989         0.9993013 
post_matrix[30,6] post_matrix[31,1] post_matrix[31,2] post_matrix[31,3] 
        0.9997074         0.9994123         0.9992770         0.9992888 
post_matrix[31,4] post_matrix[31,5] post_matrix[31,6] post_matrix[32,1] 
        0.9993967         0.9993039         0.9997086         0.9994179 
post_matrix[32,2] post_matrix[32,3] post_matrix[32,4] post_matrix[32,5] 
        0.9992717         0.9992858         0.9993945         0.9993064 
post_matrix[32,6] post_matrix[33,1] post_matrix[33,2] post_matrix[33,3] 
        0.9997097         0.9994236         0.9992667         0.9992827 
```

```
post_matrix[33,4] post_matrix[33,5] post_matrix[33,6] post_matrix[34,1]
        0.9993923         0.9993089         0.9997107         0.9994293
post_matrix[34,2] post_matrix[34,3] post_matrix[34,4] post_matrix[34,5]
        0.9992619         0.9992797         0.9993902         0.9993113
post_matrix[34,6] post_matrix[35,1] post_matrix[35,2] post_matrix[35,3]
        0.9997116         0.9994350         0.9992574         0.9992766
post_matrix[35,4] post_matrix[35,5] post_matrix[35,6] post_matrix[36,1]
        0.9993881         0.9993136         0.9997123         0.9994407
post_matrix[36,2] post_matrix[36,3] post_matrix[36,4] post_matrix[36,5]
        0.9992531         0.9992734         0.9993860         0.9993159
post_matrix[36,6] post_matrix[37,1] post_matrix[37,2] post_matrix[37,3]
        0.9997129         0.9994464         0.9992490         0.9992703
post_matrix[37,4] post_matrix[37,5] post_matrix[37,6] post_matrix[38,1]
        0.9993839         0.9993181         0.9997134         0.9994521
post_matrix[38,2] post_matrix[38,3] post_matrix[38,4] post_matrix[38,5]
        0.9992452         0.9992672         0.9993819         0.9993202
post_matrix[38,6] post_matrix[39,1] post_matrix[39,2] post_matrix[39,3]
        0.9997137         0.9994578         0.9992416         0.9992641
post_matrix[39,4] post_matrix[39,5] post_matrix[39,6] post_matrix[40,1]
        0.9993799         0.9993223         0.9997140         0.9994635
post_matrix[40,2] post_matrix[40,3] post_matrix[40,4] post_matrix[40,5]
        0.9992382         0.9992609         0.9993780         0.9993243
post_matrix[40,6] post_matrix[41,1] post_matrix[41,2] post_matrix[41,3]
        0.9997141         0.9994692         0.9992350         0.9992578
post_matrix[41,4] post_matrix[41,5] post_matrix[41,6] post_matrix[42,1]
        0.9993760         0.9993262         0.9997142         0.9994748
post_matrix[42,2] post_matrix[42,3] post_matrix[42,4] post_matrix[42,5]
        0.9992320         0.9992547         0.9993741         0.9993281
post_matrix[42,6] post_matrix[43,1] post_matrix[43,2] post_matrix[43,3]
        0.9997141         0.9994804         0.9992292         0.9992516
post_matrix[43,4] post_matrix[43,5] post_matrix[43,6] post_matrix[44,1]
        0.9993723         0.9993299         0.9997140         0.9994860
post_matrix[44,2] post_matrix[44,3] post_matrix[44,4] post_matrix[44,5]
        0.9992266         0.9992486         0.9993704         0.9993317
post_matrix[44,6] post_matrix[45,1] post_matrix[45,2] post_matrix[45,3]
        0.9997137         0.9994915         0.9992242         0.9992455
post_matrix[45,4] post_matrix[45,5] post_matrix[45,6] post_matrix[46,1]
        0.9993686         0.9993333         0.9997134         0.9994970
post_matrix[46,2] post_matrix[46,3] post_matrix[46,4] post_matrix[46,5]
        0.9992220         0.9992425         0.9993669         0.9993350
post_matrix[46,6] post_matrix[47,1] post_matrix[47,2] post_matrix[47,3]
        0.9997130         0.9995024         0.9992199         0.9992395
post_matrix[47,4] post_matrix[47,5] post_matrix[47,6] post_matrix[48,1]
        0.9993652         0.9993365         0.9997125         0.9995078
post_matrix[48,2] post_matrix[48,3] post_matrix[48,4] post_matrix[48,5]
        0.9992181         0.9992365         0.9993635         0.9993380
post_matrix[48,6] post_matrix[49,1] post_matrix[49,2] post_matrix[49,3]
        0.9997119         0.9995132         0.9992164         0.9992336
post_matrix[49,4] post_matrix[49,5] post_matrix[49,6] post_matrix[50,1]
        0.9993618         0.9993395         0.9997113         0.9995184
post_matrix[50,2] post_matrix[50,3] post_matrix[50,4] post_matrix[50,5]
        0.9992149         0.9992307         0.9993602         0.9993409
post_matrix[50,6] post_matrix[51,1] post_matrix[51,2] post_matrix[51,3]
        0.9997106         0.9995237         0.9992135         0.9992278
```

```
post_matrix[51,4] post_matrix[51,5] post_matrix[51,6] post_matrix[52,1]
        0.9993586         0.9993422         0.9997099         0.9995288
post_matrix[52,2] post_matrix[52,3] post_matrix[52,4] post_matrix[52,5]
        0.9992123         0.9992249         0.9993571         0.9993435
post_matrix[52,6] post_matrix[53,1] post_matrix[53,2] post_matrix[53,3]
        0.9997091         0.9995339         0.9992112         0.9992221
post_matrix[53,4] post_matrix[53,5] post_matrix[53,6] post_matrix[54,1]
        0.9993556         0.9993447         0.9997082         0.9995390
post_matrix[54,2] post_matrix[54,3] post_matrix[54,4] post_matrix[54,5]
        0.9992102         0.9992194         0.9993541         0.9993459
post_matrix[54,6] post_matrix[55,1] post_matrix[55,2] post_matrix[55,3]
        0.9997073         0.9995439         0.9992094         0.9992166
post_matrix[55,4] post_matrix[55,5] post_matrix[55,6] post_matrix[56,1]
        0.9993526         0.9993470         0.9997063         0.9995488
post_matrix[56,2] post_matrix[56,3] post_matrix[56,4] post_matrix[56,5]
        0.9992087         0.9992139         0.9993512         0.9993480
post_matrix[56,6] post_matrix[57,1] post_matrix[57,2] post_matrix[57,3]
        0.9997053         0.9995537         0.9992082         0.9992113
post_matrix[57,4] post_matrix[57,5] post_matrix[57,6] post_matrix[58,1]
        0.9993499         0.9993491         0.9997042         0.9995584
post_matrix[58,2] post_matrix[58,3] post_matrix[58,4] post_matrix[58,5]
        0.9992077         0.9992087         0.9993485         0.9993500
post_matrix[58,6] post_matrix[59,1] post_matrix[59,2] post_matrix[59,3]
        0.9997032         0.9995631         0.9992074         0.9992061
post_matrix[59,4] post_matrix[59,5] post_matrix[59,6] post_matrix[60,1]
        0.9993472         0.9993510         0.9997020         0.9995678
post_matrix[60,2] post_matrix[60,3] post_matrix[60,4] post_matrix[60,5]
        0.9992072         0.9992036         0.9993459         0.9993519
post_matrix[60,6] post_matrix[61,1] post_matrix[61,2] post_matrix[61,3]
        0.9997009         0.9995723         0.9992071         0.9992011
post_matrix[61,4] post_matrix[61,5] post_matrix[61,6] post_matrix[62,1]
        0.9993447         0.9993527         0.9996997         0.9995768
post_matrix[62,2] post_matrix[62,3] post_matrix[62,4] post_matrix[62,5]
        0.9992070         0.9991986         0.9993435         0.9993535
post_matrix[62,6] post_matrix[63,1] post_matrix[63,2] post_matrix[63,3]
        0.9996985         0.9995812         0.9992071         0.9991962
post_matrix[63,4] post_matrix[63,5] post_matrix[63,6] post_matrix[64,1]
        0.9993423         0.9993543         0.9996972         0.9995855
post_matrix[64,2] post_matrix[64,3] post_matrix[64,4] post_matrix[64,5]
        0.9992073         0.9991939         0.9993411         0.9993550
post_matrix[64,6] post_matrix[65,1] post_matrix[65,2] post_matrix[65,3]
        0.9996960         0.9995898         0.9992075         0.9991916
post_matrix[65,4] post_matrix[65,5] post_matrix[65,6] post_matrix[66,1]
        0.9993400         0.9993557         0.9996947         0.9995940
post_matrix[66,2] post_matrix[66,3] post_matrix[66,4] post_matrix[66,5]
        0.9992078         0.9991893         0.9993389         0.9993563
post_matrix[66,6] post_matrix[67,1] post_matrix[67,2] post_matrix[67,3]
        0.9996934         0.9995981         0.9992082         0.9991871
post_matrix[67,4] post_matrix[67,5] post_matrix[67,6] post_matrix[68,1]
        0.9993378         0.9993569         0.9996921         0.9996022
post_matrix[68,2] post_matrix[68,3] post_matrix[68,4] post_matrix[68,5]
        0.9992087         0.9991849         0.9993368         0.9993575
post_matrix[68,6] post_matrix[69,1] post_matrix[69,2] post_matrix[69,3]
        0.9996907         0.9996061         0.9992092         0.9991827
```

```
post_matrix[69,4] post_matrix[69,5] post_matrix[69,6] post_matrix[70,1]
        0.9993358         0.9993581         0.9996894         0.9996100
post_matrix[70,2] post_matrix[70,3] post_matrix[70,4] post_matrix[70,5]
        0.9992098         0.9991806         0.9993348         0.9993586
post_matrix[70,6] post_matrix[71,1] post_matrix[71,2] post_matrix[71,3]
        0.9996880         0.9996139         0.9992105         0.9991785
post_matrix[71,4] post_matrix[71,5] post_matrix[71,6] post_matrix[72,1]
        0.9993339         0.9993591         0.9996866         0.9996177
post_matrix[72,2] post_matrix[72,3] post_matrix[72,4] post_matrix[72,5]
        0.9992112         0.9991765         0.9993329         0.9993595
post_matrix[72,6] post_matrix[73,1] post_matrix[73,2] post_matrix[73,3]
        0.9996853         0.9996214         0.9992119         0.9991745
post_matrix[73,4] post_matrix[73,5] post_matrix[73,6] post_matrix[74,1]
        0.9993320         0.9993600         0.9996839         0.9996250
post_matrix[74,2] post_matrix[74,3] post_matrix[74,4] post_matrix[74,5]
        0.9992128         0.9991726         0.9993311         0.9993604
post_matrix[74,6] post_matrix[75,1] post_matrix[75,2] post_matrix[75,3]
        0.9996825         0.9996286         0.9992136         0.9991706
post_matrix[75,4] post_matrix[75,5] post_matrix[75,6] post_matrix[76,1]
        0.9993303         0.9993607         0.9996811         0.9996321
post_matrix[76,2] post_matrix[76,3] post_matrix[76,4] post_matrix[76,5]
        0.9992145         0.9991688         0.9993294         0.9993611
post_matrix[76,6] post_matrix[77,1] post_matrix[77,2] post_matrix[77,3]
        0.9996797         0.9996355         0.9992155         0.9991669
post_matrix[77,4] post_matrix[77,5] post_matrix[77,6] post_matrix[78,1]
        0.9993286         0.9993614         0.9996783         0.9996389
post_matrix[78,2] post_matrix[78,3] post_matrix[78,4] post_matrix[78,5]
        0.9992165         0.9991651         0.9993278         0.9993617
post_matrix[78,6] post_matrix[79,1] post_matrix[79,2] post_matrix[79,3]
        0.9996769         0.9996422         0.9992175         0.9991634
post_matrix[79,4] post_matrix[79,5] post_matrix[79,6] post_matrix[80,1]
        0.9993270         0.9993620         0.9996755         0.9996454
post_matrix[80,2] post_matrix[80,3] post_matrix[80,4] post_matrix[80,5]
        0.9992186         0.9991616         0.9993263         0.9993622
post_matrix[80,6] post_matrix[81,1] post_matrix[81,2] post_matrix[81,3]
        0.9996741         0.9996486         0.9992197         0.9991599
post_matrix[81,4] post_matrix[81,5] post_matrix[81,6] post_matrix[82,1]
        0.9993256         0.9993625         0.9996727         0.9996518
post_matrix[82,2] post_matrix[82,3] post_matrix[82,4] post_matrix[82,5]
        0.9992208         0.9991583         0.9993249         0.9993627
post_matrix[82,6] post_matrix[83,1] post_matrix[83,2] post_matrix[83,3]
        0.9996713         0.9996548         0.9992219         0.9991566
post_matrix[83,4] post_matrix[83,5] post_matrix[83,6] post_matrix[84,1]
        0.9993242         0.9993629         0.9996699         0.9996578
post_matrix[84,2] post_matrix[84,3] post_matrix[84,4] post_matrix[84,5]
        0.9992231         0.9991551         0.9993235         0.9993631
post_matrix[84,6] post_matrix[85,1] post_matrix[85,2] post_matrix[85,3]
        0.9996685         0.9996608         0.9992243         0.9991535
post_matrix[85,4] post_matrix[85,5] post_matrix[85,6] post_matrix[86,1]
        0.9993228         0.9993632         0.9996671         0.9996637
post_matrix[86,2] post_matrix[86,3] post_matrix[86,4] post_matrix[86,5]
        0.9992255         0.9991520         0.9993222         0.9993634
post_matrix[86,6] post_matrix[87,1] post_matrix[87,2] post_matrix[87,3]
        0.9996657         0.9996665         0.9992268         0.9991505
```

```
 post_matrix[87,4]  post_matrix[87,5]  post_matrix[87,6]  post_matrix[88,1]
         0.9993216          0.9993635          0.9996644          0.9996693
 post_matrix[88,2]  post_matrix[88,3]  post_matrix[88,4]  post_matrix[88,5]
         0.9992281          0.9991490          0.9993210          0.9993636
 post_matrix[88,6]  post_matrix[89,1]  post_matrix[89,2]  post_matrix[89,3]
         0.9996630          0.9996720          0.9992293          0.9991476
 post_matrix[89,4]  post_matrix[89,5]  post_matrix[89,6]  post_matrix[90,1]
         0.9993204          0.9993637          0.9996616          0.9996747
 post_matrix[90,2]  post_matrix[90,3]  post_matrix[90,4]  post_matrix[90,5]
         0.9992306          0.9991462          0.9993198          0.9993638
 post_matrix[90,6]  post_matrix[91,1]  post_matrix[91,2]  post_matrix[91,3]
         0.9996603          0.9996773          0.9992319          0.9991448
 post_matrix[91,4]  post_matrix[91,5]  post_matrix[91,6]  post_matrix[92,1]
         0.9993193          0.9993639          0.9996589          0.9996799
 post_matrix[92,2]  post_matrix[92,3]  post_matrix[92,4]  post_matrix[92,5]
         0.9992333          0.9991434          0.9993187          0.9993639
 post_matrix[92,6]  post_matrix[93,1]  post_matrix[93,2]  post_matrix[93,3]
         0.9996576          0.9996824          0.9992346          0.9991421
 post_matrix[93,4]  post_matrix[93,5]  post_matrix[93,6]  post_matrix[94,1]
         0.9993182          0.9993640          0.9996563          0.9996849
 post_matrix[94,2]  post_matrix[94,3]  post_matrix[94,4]  post_matrix[94,5]
         0.9992359          0.9991408          0.9993177          0.9993640
 post_matrix[94,6]  post_matrix[95,1]  post_matrix[95,2]  post_matrix[95,3]
         0.9996550          0.9996873          0.9992373          0.9991396
 post_matrix[95,4]  post_matrix[95,5]  post_matrix[95,6]  post_matrix[96,1]
         0.9993172          0.9993640          0.9996536          0.9996897
 post_matrix[96,2]  post_matrix[96,3]  post_matrix[96,4]  post_matrix[96,5]
         0.9992387          0.9991383          0.9993167          0.9993641
 post_matrix[96,6]  post_matrix[97,1]  post_matrix[97,2]  post_matrix[97,3]
         0.9996523          0.9996921          0.9992400          0.9991371
 post_matrix[97,4]  post_matrix[97,5]  post_matrix[97,6]  post_matrix[98,1]
         0.9993163          0.9993641          0.9996510          0.9996943
 post_matrix[98,2]  post_matrix[98,3]  post_matrix[98,4]  post_matrix[98,5]
         0.9992414          0.9991359          0.9993158          0.9993640
 post_matrix[98,6]  post_matrix[99,1]  post_matrix[99,2]  post_matrix[99,3]
         0.9996498          0.9996966          0.9992428          0.9991347
 post_matrix[99,4]  post_matrix[99,5]  post_matrix[99,6] post_matrix[100,1]
         0.9993154          0.9993640          0.9996485          0.9996988
post_matrix[100,2] post_matrix[100,3] post_matrix[100,4] post_matrix[100,5]
         0.9992442          0.9991336          0.9993150          0.9993640
post_matrix[100,6] post_matrix[101,1] post_matrix[101,2] post_matrix[101,3]
         0.9996472          0.9997010          0.9992455          0.9991325
post_matrix[101,4] post_matrix[101,5] post_matrix[101,6]               lp__
         0.9993145          0.9993640          0.9996460          1.0033983
```