Lab Report 1

Mitheysh Asokan, Jason (Eungjoo) Kim

Question 1: Linear Regression

Chosen Dataset: spotify_songs

Output Variable: danceability

Input Variables: playlist_genre , speechiness, energy, duration_ms

1.1. (10 pts)

Give basic insights into your numeric variable you have picked as output variable using one categorical variable you selected.

- What are the min / max values and median of the output variable, Y?

```
songs <- read.csv('spotify_songs.csv')

max(songs$danceability)
```

```
## [1] 0.983
```

```
min(songs$danceability)
```

```
## [1] 0
```

```
median(songs$danceability)
```

```
## [1] 0.672
```

- What is median of the output value among different classes of the categorical variable you picked? You must use `group_by` and `summarize` functions.

```
group_by(songs, playlist_genre) %>%
  summarise(median.danceability=median(danceability))
```

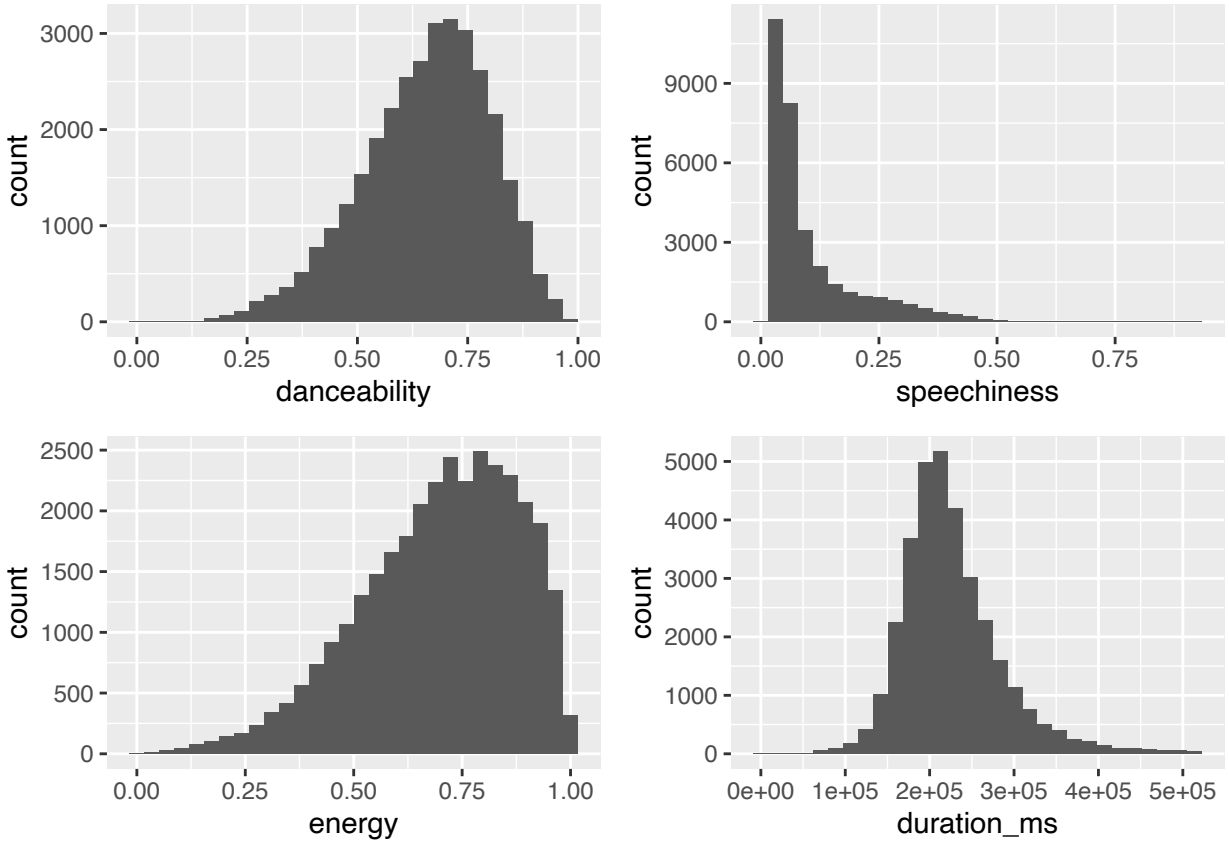
```
## # A tibble: 6 x 2
##   playlist_genre median.danceability
## * <chr>                        <dbl>
## 1 edm                          0.659
## 2 latin                        0.729
## 3 pop                          0.652
## 4 r&b                          0.689
## 5 rap                          0.737
## 6 rock                         0.523
```

1.2. (10 pts)

Visualize the variables you selected.

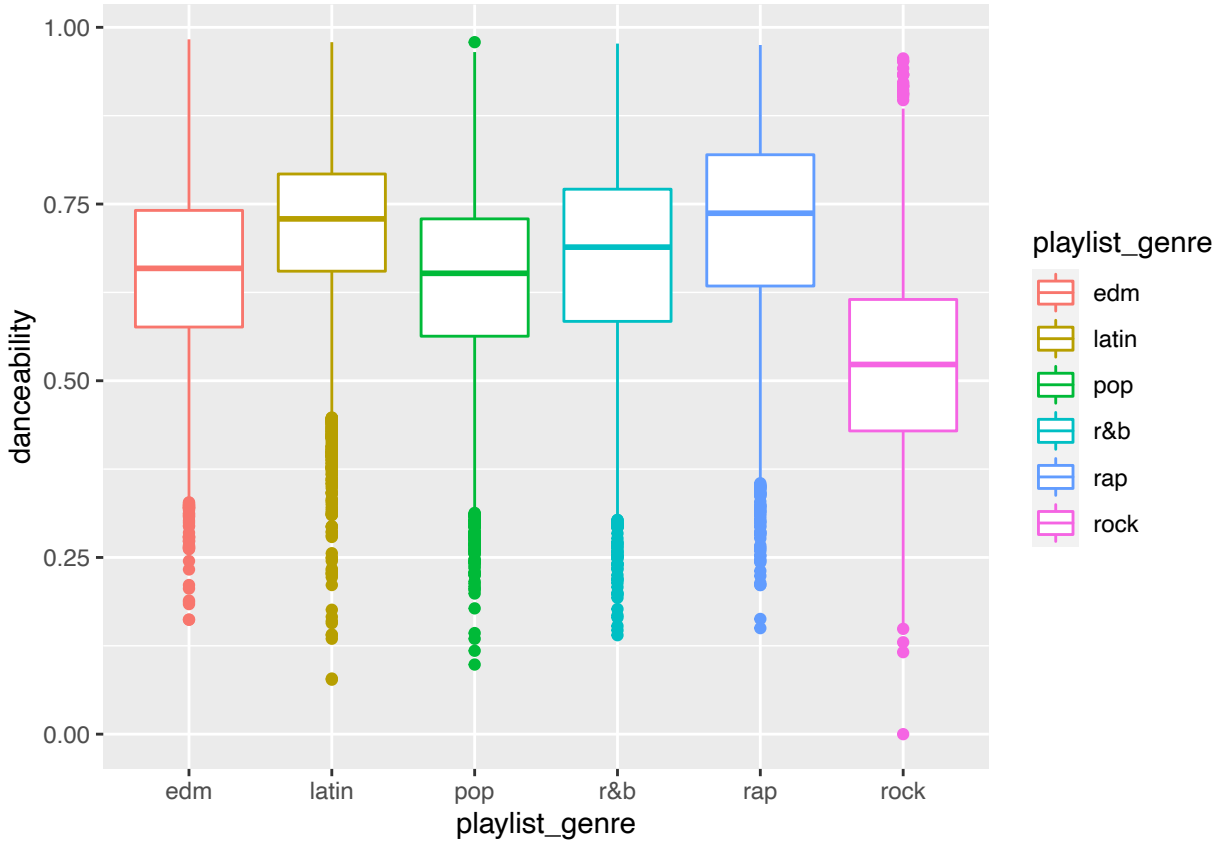
- Draw histogram of the numeric variables you selected.

```
g1 <- ggplot(data = songs) +
  geom_histogram(aes(x = danceability))

g2 <- ggplot(data = songs) +
  geom_histogram(aes(x = speechiness))

g3 <- ggplot(data = songs) +
  geom_histogram(aes(x = energy))

g4 <- ggplot(data = songs) +
  geom_histogram(aes(x = duration_ms))

grid.arrange(g1,g2,g3,g4, ncol=2)
```



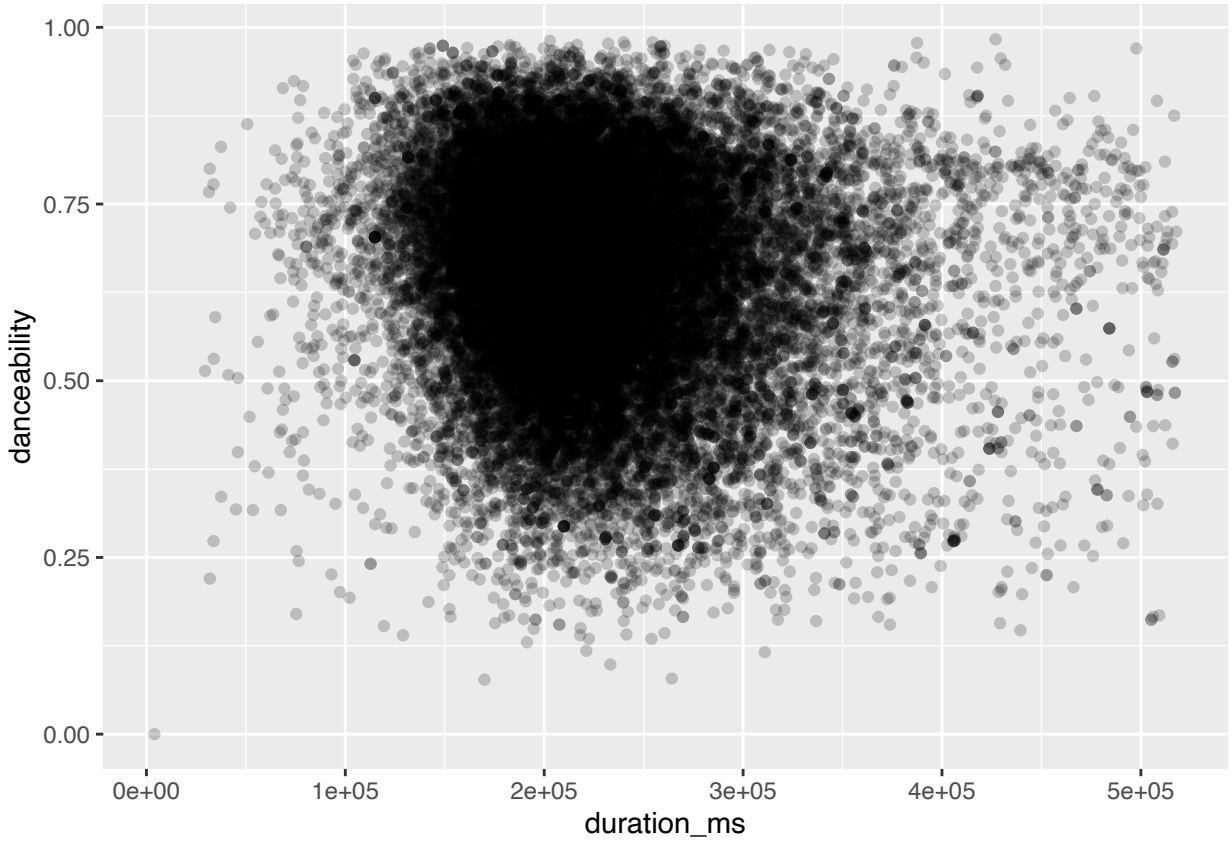
- Draw distribution of the output variable Y with respect to the different classes of your categorical variable. The plot must somehow show the distributional differences among different classes. You can use boxplot, histogram, or other visuals (e.g. density ringes).

```
songs$playlist_genre <- as.factor(songs$playlist_genre)

ggplot(songs, aes(x=playlist_genre, y=danceability, colour=playlist_genre)) +
  geom_boxplot()
```



- Draw scatter plot between one of your numeric inputs and the output variable. Discuss whether the plot indicate a relation, if it is linear, if there are outliers? Feel free to remove the outlier. Feel free to transform the data.

```
ggplot(songs, aes(x=duration_ms, y=danceability)) +
  geom_point(alpha=0.2)
```



The danceability of a song is clearly dependent to its duration. The relationship is evident, since the dancebility of a track is falling when its duration is rising.

The relationship is somewhat linear based on the scatterplot.

1.3. (15 pts)

Using the all dataset, fit a regression:

1. Using the one numeric input variable fit a simple regression model.

- Write down the model.
- Fit the regression line.
- Summarize the output.
- Plot the input and output variables in a scatter plot and add the predicted values as a line.
- Interpret the results. Is it a good fit? Is your input variable good in explaining the outputs?

$HDanceability_i = \beta_0 + \beta_1 Duration_i + \epsilon_i$

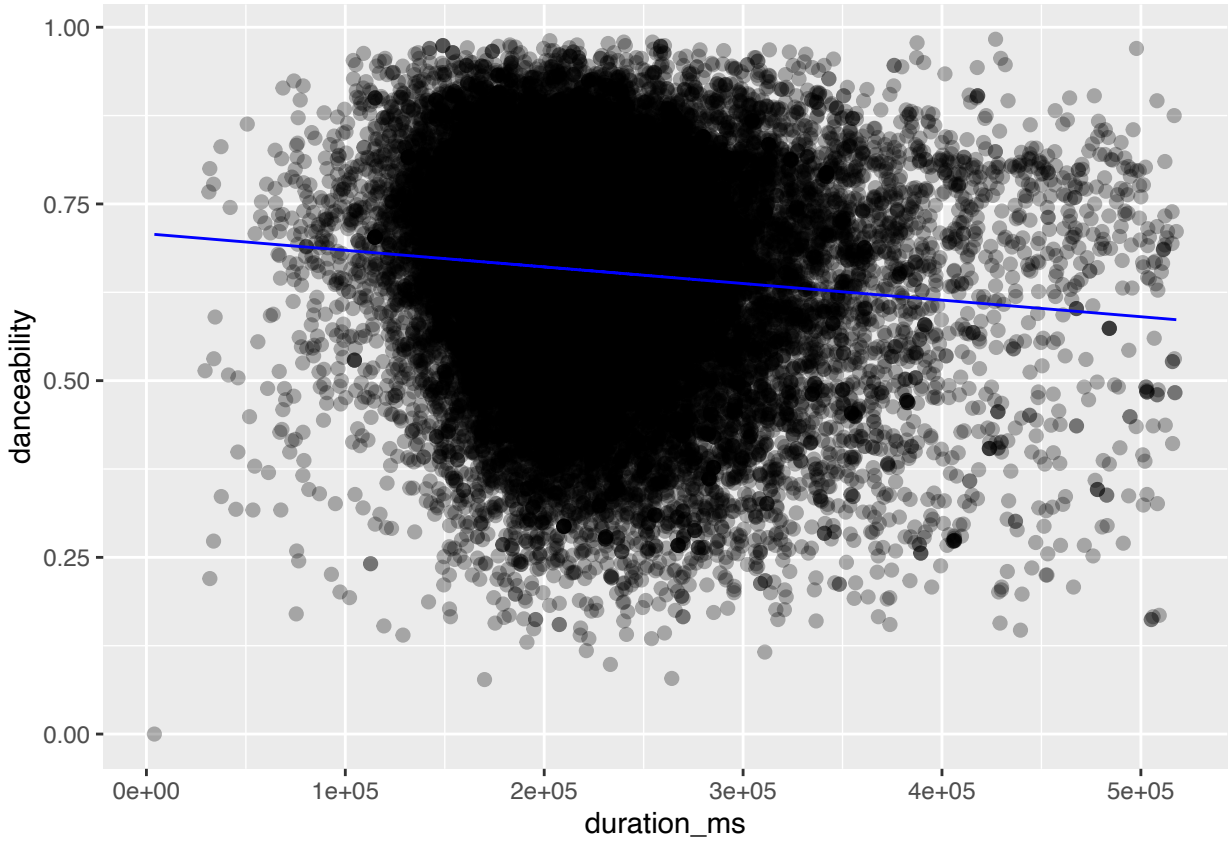
```
fit1 <- lm(danceability ~ duration_ms, data = songs)
```

```
summary(fit1)
```

```
##
```

```
## Call:
## lm(formula = danceability ~ duration_ms, data = songs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.70695 -0.09228  0.01505  0.10441  0.37904
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.079e-01  3.111e-03  227.52   <2e-16 ***
## duration_ms -2.349e-07  1.332e-08  -17.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1444 on 32831 degrees of freedom
## Multiple R-squared:  0.009385,   Adjusted R-squared:  0.009355
## F-statistic: 311.1 on 1 and 32831 DF,  p-value: < 2.2e-16
```

```
preds <- predict(fit1)
ggplot(songs, aes(x=duration_ms, y=danceability)) +
  geom_point(alpha=.3, size=2) +
  geom_line(aes(y=preds), colour="blue")
```



The line is appropriate fit (very low Mean Root Squared), that is downard sloping. Which means that danceability gradually falls as the duration of the song increases. This explanation is in line with how

exhausted a dancer is getting with time. The longer the song, the more exhausted the dancer gets, which leads to stoppage.

According to the summary, the data for danceability is strongly correlated with duration.

The p-values are very close to zero and they are significant (due to the ***).

The f-stat is also very high for the degrees of freedom used.

The RMSE is also very low.

2. Using all your input variables, fit a multiple linear regression model

    - Write down the model
    - Fit the regression line and summarize the output
    - Interpret the results. Is it a good fit? Are the input variables good in explaining the outputs?

$HDanceability_i = \beta_0 + \beta_1 Duration_i + \beta_2 Energy_i + \beta_3 Speechiness_i + \beta_4 PlaylistGenre_i + \epsilon_i$

```
fit2 <- lm(danceability ~ duration_ms + energy + speechiness + playlist_genre, data = songs)
```

```
summary(fit2)
```

```
##
## Call:
## lm(formula = danceability ~ duration_ms + energy + speechiness +
##     playlist_genre, data = songs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.65144 -0.07799  0.01334  0.09187  0.42348
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           6.869e-01  4.688e-03 146.522  < 2e-16 ***
## duration_ms          -5.601e-08  1.228e-08  -4.561 5.12e-06 ***
## energy               -3.227e-02  4.284e-03  -7.533 5.09e-14 ***
## speechiness           7.497e-02  7.964e-03   9.413  < 2e-16 ***
## playlist_genrelatin   5.369e-02  2.505e-03  21.437  < 2e-16 ***
## playlist_genrepop    -1.833e-02  2.464e-03  -7.438 1.04e-13 ***
## playlist_genrer&b     6.898e-03  2.622e-03   2.631  0.00851 **
## playlist_genrerap     4.964e-02  2.650e-03  18.729  < 2e-16 ***
## playlist_genrerock   -1.331e-01  2.540e-03 -52.400  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1301 on 32824 degrees of freedom
## Multiple R-squared:  0.196,  Adjusted R-squared:  0.1958
## F-statistic:  1000 on 8 and 32824 DF,  p-value: < 2.2e-16
```

The fit is appropriate (low Root Mean Squared, but it is getting better as we add more input variables), which means that danceability related to duration, energy, speechiness, and genre of the song.

The p-values are very close to zero and they are significant (due to the ***). Except for the R&B genre, which only score 2 asterisks for coefficient significance. This is expected as the genre is not intended for dancing.

The f-stat is also very high for the degrees of freedom used.

The RMSE is also very low.

3. Now, do the same things as you did, but this time add an interaction between one categorical and one numeric variable.
    - Write down the model, fit to the data, summarize and interpret the results.

$HDanceability_i = \beta_0 + \beta_1 Duration_i + \beta_2 Energy_i + \beta_3 Speechiness_i + \beta_4 PlaylistGenre_i + \beta_5 PlaylistGenre_i Speechiness_i + \epsilon_i$

```
fit3 <- lm(danceability ~ duration_ms + energy + speechiness + playlist_genre + playlist_genre:spe
```

```
summary(fit3)
```

```
##
## Call:
## lm(formula = danceability ~ duration_ms + energy + speechiness +
##     playlist_genre + playlist_genre:speechiness, data = songs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.64526 -0.07738  0.01309  0.09095  0.44727
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      6.733e-01  5.112e-03 131.711  < 2e-16 ***
## duration_ms                     -5.461e-08  1.229e-08  -4.444 8.85e-06 ***
## energy                          -2.568e-02  4.317e-03  -5.948 2.74e-09 ***
## speechiness                      1.676e-01  2.352e-02   7.127 1.05e-12 ***
## playlist_genrelatin              5.872e-02  3.846e-03  15.267  < 2e-16 ***
## playlist_genrepop               -1.301e-02  3.715e-03  -3.503 0.000461 ***
## playlist_genrer&b                7.849e-03  3.838e-03   2.045 0.040839 *
## playlist_genrerap                6.902e-02  4.113e-03  16.781  < 2e-16 ***
## playlist_genrerock              -9.714e-02  4.044e-03 -24.020  < 2e-16 ***
## speechiness:playlist_genrelatin -5.729e-02  3.126e-02  -1.833 0.066859 .
## speechiness:playlist_genrepop   -4.682e-02  3.488e-02  -1.342 0.179457
## speechiness:playlist_genrer&b   -2.025e-02  2.867e-02  -0.706 0.480019
## speechiness:playlist_genrerap   -1.450e-01  2.685e-02  -5.399 6.73e-08 ***
## speechiness:playlist_genrerock  -5.696e-01  4.737e-02 -12.023  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1297 on 32819 degrees of freedom
## Multiple R-squared:  0.2007, Adjusted R-squared:  0.2004
## F-statistic: 633.8 on 13 and 32819 DF,  p-value: < 2.2e-16
```

The fit is appropriate (low Root Mean Squared, but it is getting better as we add more input variables), which means that danceability related to duration, energy, speechiness, genre of the song, and the speechiness of each genre.

The p-values are very close to zero and they are significant (due to the ***). Except for the R&B genre, which only score 1 asterisk for coefficient significance (This is expected as the genre is not intended for dancing) and the speechiness of the latin, pop, and r&b genres (Which is also expected, as the latin and pop songs tend to focus on the music instead of the lyric to get people dancing to them).

The f-stat is also very high for the degrees of freedom used.

The RMSE is also very low.

4. Which model you fit is the best in predicting the output variable? Which one is the second and third best? Rank the models based on their performance.

```
s1 <- sigma(fit1)
s2 <- sigma(fit2)
s3 <- sigma(fit3)

result <- c(s1, s2, s3)
result
```

```
## [1] 0.1444051 0.1301085 0.1297394
```

RANKING: fit3, fit2, fit1

1.4. (15 pts)

In this section, you will do the same you did in 1.3, but this time you will first split the data into train and test.

- Select seed to fix the random numbers you will generate using `set.seed(...)`.

```
set.seed(100)
```

- Split your data into test and train sets with 20/80 test-train ratio.

```
train_size <- floor(0.8 * nrow(songs))
train_inds <- sample(1:nrow(songs), size = train_size)
test_inds  <- setdiff(1:nrow(songs), train_inds)

train <- songs[ train_inds , ]
test  <- songs[ test_inds , ]
```

- Fit the model to the train set and evaluate the how well the model performed on test set.

```
fit1 <- lm(danceability ~ duration_ms, data = train)
fit2 <- lm(danceability ~ duration_ms + energy + speechiness + playlist_genre, data = train)
fit3 <- lm(danceability ~ duration_ms + energy + speechiness + playlist_genre + playlist_genre:spe

pred1 <- predict(fit1, newdata=test)
pred2 <- predict(fit2, newdata=test)
pred3 <- predict(fit3, newdata=test)
```

- Which model performed the best on test set? Rank the models based ion their performance.

```
rmse1 <- RMSE(pred1, test$danceability)
rmse2 <- RMSE(pred2, test$danceability)
rmse3 <- RMSE(pred3, test$danceability)

result <- c(rmse1,rmse2,rmse3)
result
```

```
## [1] 0.1446211 0.1300120 0.1297335
```

- Is the rank the same as the one you had in 1.3?

Yes.