Exploratory Analysis on Spotify Songs

Derian Lee

Dataset:

```
spotify_songs <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/
```

Link to the dataset: https://github.com/rfordatascience/tidytuesday/blob/master/data/2020/2020-01-21/readme.md

Part 1

Question: When looking between rap, rock and EDM, how do proportions differ between mode and genre, and how does the distribution for tempo differ across these genres?

Introduction:

The *spotify_songs* dataset contains various information about songs, such as its genre, mode, volume, valence, and much more. This dataset is subsetted into *spotify_songs1* to focus on the three genres, which will be used for this question. The question to answer is: when looking between rock, rap and EDM (1) how does the proportion of mode differ across genre, and (2) how does the distribution of tempo differ across the genres? To answer this question, several variables will be needed from the *spotify_songs1* dataset. The following variables will be taken: *playlist_genre*, *mode*, and *tempo*.

Note: the variable "mode" in this sense, refers to the musical scale used in the song. It can take two values in this dataset: 0 represents a minor (sad, negative) scale, and 1 represents the major (happy, upbeat) scale.

Approach:

DATA WRANGLING / SUMMARY TABLE:

Because we are only interested in rock, rap and EDM, the *filter()* function will first be used to keep songs within these genres only. Next, *count()* and *pivot_wider()* allows us to return mode count per genre. Then, the *mutate()* function is used to rename 1 and 0 into their respective components (major and minor). It is also used along with mathematical operators to add our columns of interest, which is the mode proportion per genre. Lastly, *select()* is used to remove the original columns, (`1` and `0`) as they have already been replaced with "major" and "minor", as described above. To answer the proportion question, a summary table was created instead of a visualization because there are categories with proportions 52 / 48. I would argue that visually, it is hard to distinguish between a 50 / 50 split and a 52 / 48 split and thus, a summary table was chosen for clarity's sake.

Note: For the second part of this question, one limitation to density plots is that they don't show the counts that make up each category, and can be misleading if the sample size is too low. Thus, this is added into our summary table to ensure that we have a sufficient sample size for each category.

PLOT:

For the second part of the question, a faceted density plot will be used. This was chosen because density plots allow us to visualize the distribution of song tempos, which we can then facet across our genres of interest. Although a histogram could also be used to answer our question, a density plot was chosen for our analysis. This is because for this question, we are more interested in the overall shape of the distribution,

rather than comparing across tempo groupings (such as comparing relative genre counts for 110 BPM vs 200 BPM). I would also argue that tempo is less of a fixed variable, and can fluctuate within a song. Thus, I would consider these cut-offs in a histogram to be arbitrary / less significant.

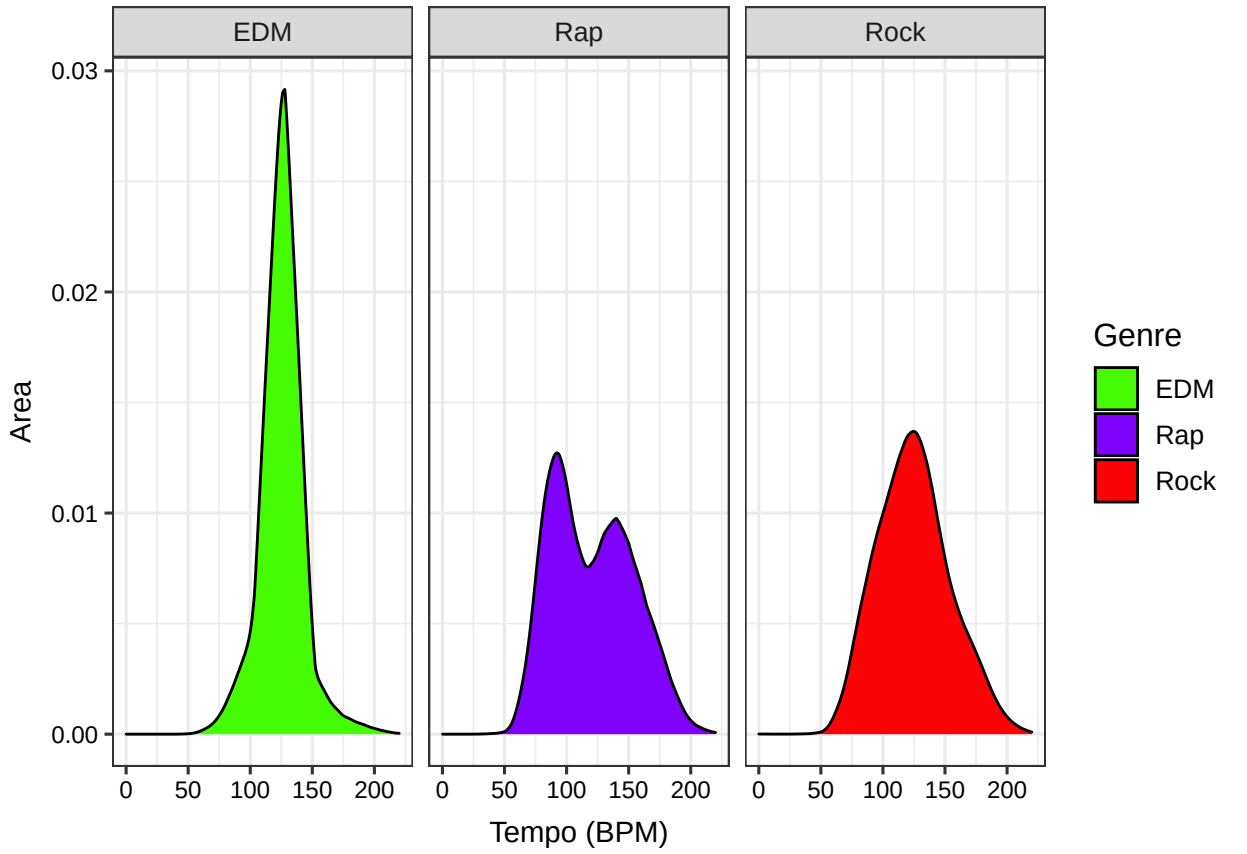
Analysis:

```
#DATA PREPARATION: Filtering for genres of interest:
spotify_songs1 <- spotify_songs %>%
  filter(playlist_genre == "rock"|
         playlist_genre == "rap"|
         playlist_genre =="edm")

#Summary Table: Proportion of Modes for each Genre
spotify_songs1 %>%
  count(playlist_genre, mode) %>% #return mode count per genre
  pivot_wider(names_from = "mode", values_from = "n") %>%
  mutate( #adding new columns
    Major = `1`,
    Minor = `0`,
    Total_Count = `1` + `0`,
    Major_Prop. = (`1` / (`1` + `0`)),
    Minor_Prop. = (`0` / (`1` + `0`))
        ) %>%
  select(-`0`,-`1`) #removing old columns
```

```
## # A tibble: 3 x 6
##   playlist_genre Major Minor Total_Count Major_Prop. Minor_Prop.
##   <chr>          <int> <int>       <int>       <dbl>       <dbl>
## 1 edm             3143  2900        6043       0.520       0.480
## 2 rap             2996  2750        5746       0.521       0.479
## 3 rock            3467  1484        4951       0.700       0.300
```

```
#Visualization: tempo distribution between genres
ggplot(spotify_songs1, aes(tempo, fill = playlist_genre)) +
  geom_density(
    bw = 10, kernel = "triangular", color = "black") +
  facet_wrap(vars(playlist_genre), ncol = 3, #facet by genre
             labeller = as_labeller(
               c('edm' = "EDM", 'rap' = "Rap", 'rock' = "Rock")) #renaming labels
             ) +
  scale_x_continuous ( #renaming x axis, setting breaks
    name = "Tempo (BPM)",
    breaks = c(0, 50, 100, 150, 200)
  ) +
  scale_y_continuous ( #renaming y axis
    name = "Area"
  ) +
  scale_fill_manual( #renaming scale and scale labels, setting colors for scale
    name = "Genre",
    labels = c('edm' = "EDM", 'rap' = "Rap", 'rock' = "Rock"),
    values=c("#45fc03", "#7f03fc", "#fc0307")
    ) +
  theme_bw(12) +
  theme(
```

```
  axis.text = element_text(size = 9, color = "black"), #resizing and recoloring axes labels
  axis.title.x = element_text(size = 11, vjust = -0.5), #resizing, shifting axis title
  axis.title.y = element_text(size = 11, vjust = 2) #resizing, shifting axis title
)
```



Discussion:

The first part of the question (summary table) looks at the proportion of mode across genre. The summary table reveals that mode proportion is nearly identical for EDM and rap, and both genres seem to have less preference for mode (close to 50:50). However, Rock music differs considerably in proportion from these two genres, and seems to prefer major key over minor key. Lastly, all genres are more commonly written in major key than minor, although barely for EDM and rap. One reason rock music deviates from EDM and Rap might be because rock has origins in the Blues genre. That is, the Blues is largely in major (happy) key, arguably because it arose from a difficult sociopolitical context and served as a form of escape at the time.

The second part of the question (plot) looks at tempo distribution across genres. The plot reveals that, visually speaking, EDM and Rock music appears to be normally distributed. Although both genres have most songs written between 100-150 BPM, this is especially true for EDM. On the other hand, rap music appears to follow a bimodal distribution, with most songs written around 75-100 BPM and 130-150 BPM. Thus, roughly speaking, a tempo from 100-150 BPM should at least be suitable for all three genres. One reason why rap may be bimodally distributed is due to its nature. I would speculate that, since rap is predominantly spoken word, it finds popularity in either profound lyrics (which are more geared for slower tempo for clarity of message) or faster lyrics that are simply made to impress (songs like Eminem's *Rap God*, for example).

Part 2

Question:

Within the three LoFi playlists, is there a linear relationship between valence and danceability, and if so, how do these linear relationships differ among the playlists?

Introduction:

The *spotify_songs* dataset is referenced again for Part 2, and information about the dataset is described above. This dataset is then subsetted into *spotify_songs2* to focus on LoFi songs, which will be used in this question. The question to answer is, for the three LoFi playlists (1) is there a linear relationship between valence and danceability, and if so, (2) how do these linear relationships differ among the playlists? To answer this question, several variables from the dataset will be needed. The following variables will be taken from the dataset: *playlist_name*, *danceability*, and *valence*.

Note: The variable *valence* measures how positive a song is, and *danceability* measures how suitable a song is for dancing (via TidyTuesday description).

Approach:

GENERAL DATA WRANGLING: Because we are only interested in LoFi songs, the *filter()* function will be used to keep songs in LoFi playlists only. However, the playlist named "House/Electro/Progressive/Disco/Lofi/Synthwave" is removed, as this playlist includes other genres besides LoFi.

Note: To check if there are duplicate songs within multiple playlists, we use the duplicate() function. This was done because duplicate songs could interfere with our conclusion.

LINEAR MODEL DATA WRANGLING: First we will use the *nest()* function to group all of the variables by their respective playlist_name. Next, we'll use the *mutate()* function to create a column called "fit", which contains the output of the applied LM formula to each nested data table. Next, we'll use the *glance()* function on the "fit" column to provide a computer summary table of each fit object. Lastly, we'll use *select()* to only choose our columns of interest, and *unnest()* to retrieve the summary table for the three playlists. We then pass this information into label_data, and superimpose label_data on our linear model to create our first plot. We repeat this process for the second plot, with the two main differences being (1) *tidy()* is used instead of *glance()* and (2) we don't pass into label_data, but instead plug the summary table straight into our *stat_dist_dotsinterval()* function.

PLOT: For the first part of the question, we will use a linear regression plot, faceted across the three LoFi playlists. Although a faceted scatter plot may have been used as well, the human eye cannot always detect correlation. Sometimes we may, but the correlation might be insignificant. A linear model provides a p-value which gives us better statistical evidence to believe a relationship exists. For the second part of the question, an error-bar visualization will be used (with quantiles). Although we could have just relied on the linear regression to see how slopes visually differ, this visualization makes it easier to see differences in slopes, rather than having to approximate angles from a graph. Furthermore, quantiles are used to visualize how certain we are in our slope estimates. Although a half-eye or gradient interval could be used to achieve the same purpose, a quantile dotplot was chosen because it is easiest to understand for the human eye / brain. For example, we can deduce the probability of being x units from the slope estimate by using simple fractional analysis on dots. This same analysis couldn't be done on the other two graphs and thus, a quantile dotplot was chosen for better human interpretability.

Analysis:

```
#DATA PREPARATION: Lofi songs only
spotify_songs2 <- spotify_songs %>%
  filter(str_detect(playlist_name, 'Lofi|Lo-Fi') &
         playlist_name != "House/Electro/Progressive/Disco/Lofi/Synthwave")
  #Remove this playlist, as some genres here are highly different from Lofi
```

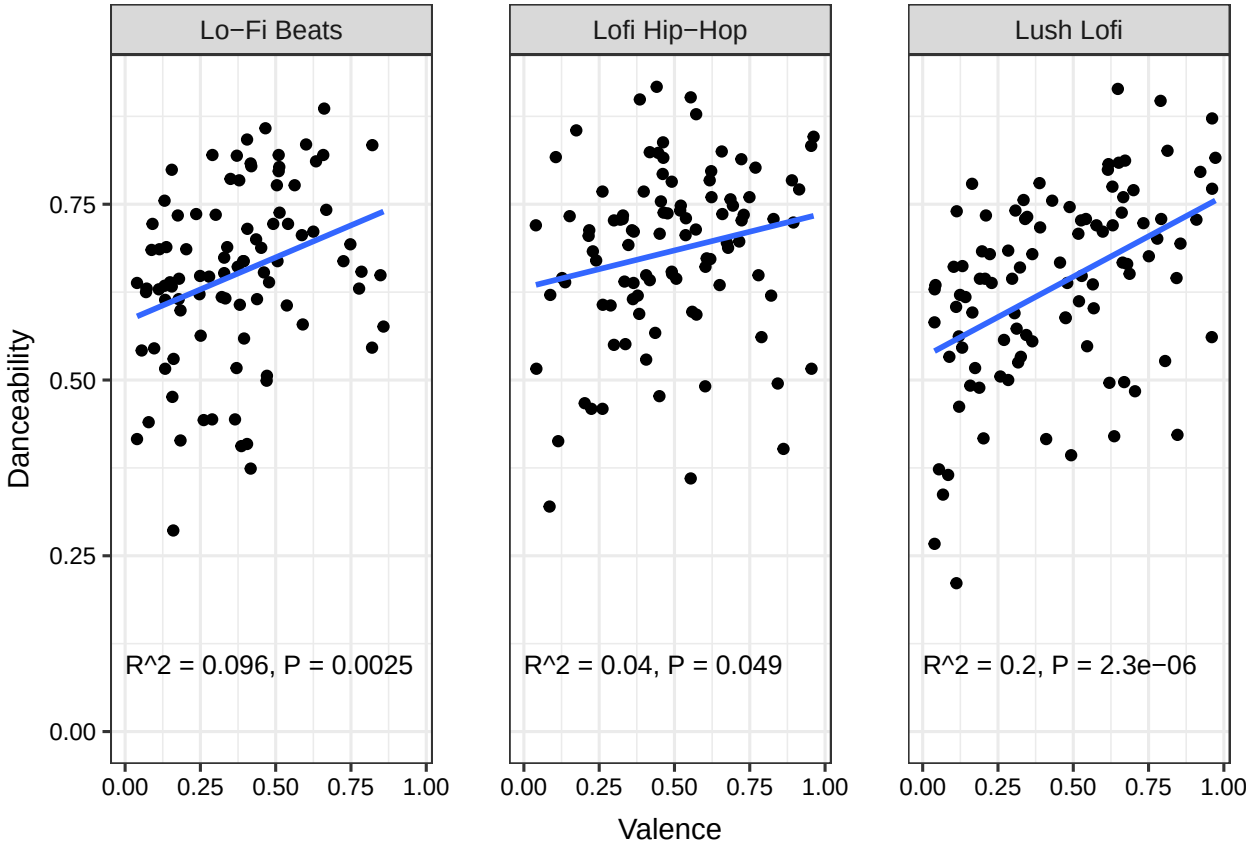
```
duplicated(spotify_songs2) #checking for duplicates: duplicate songs could interfere with conclusion
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [157] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [169] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [181] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [193] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [205] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [217] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [229] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [241] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [253] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [265] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [277] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [289] FALSE FALSE
```

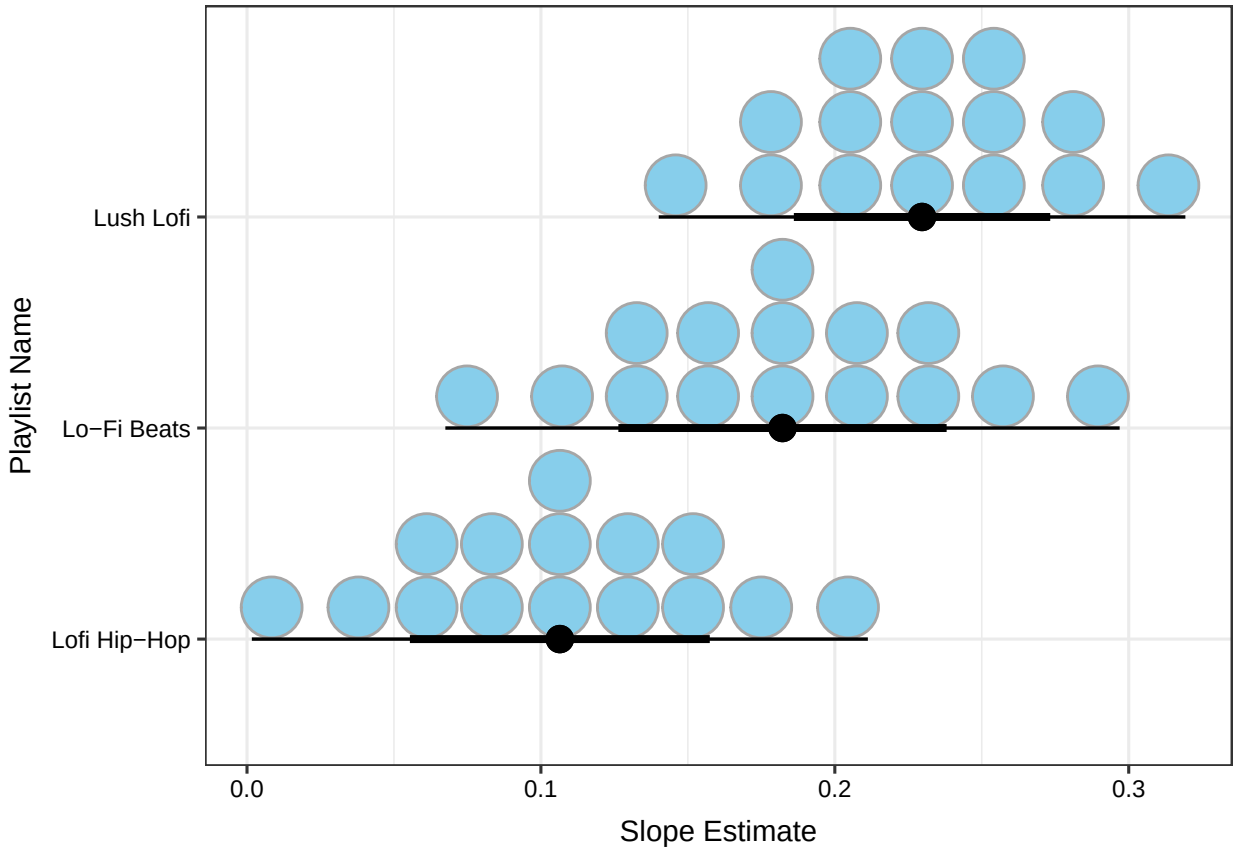
```
#no duplicates found


#Setting up linear model
lm_summary <- spotify_songs2 %>%
  nest(data = -playlist_name) %>% #group variables by playlist_name
  mutate( #create a column called "fit" and apply lm formula to each data table for playlist_name
    fit = map(data, ~lm(danceability ~ valence, data = .x)),
    glance_out = map(fit, glance) #apply glance on each fit object
  ) %>%
  select(playlist_name, glance_out) %>%
  unnest(cols = glance_out) #retrieve summary table for the three playlists

#Information for labels
label_data <- lm_summary %>%
  mutate(
    rsqr = signif(r.squared, 2),  #two significant digits for r squared value
    pval = signif(p.value, 2),
    label = glue("R^2 = {rsqr}, P = {pval}"),
    valence = 0, danceability = 0
  ) %>%
  select(playlist_name, label, valence, danceability)

#Linear model
ggplot(spotify_songs2, aes(valence, danceability)) +
```

```
  geom_point() +
  geom_text(
    data = label_data, aes(label = label),
    size = 10/.pt, vjust = -3, hjust = 0  #adjusting text size and position
  ) +
  geom_smooth(method = "lm", se = FALSE) +
  facet_wrap(vars(playlist_name)) +
  scale_x_continuous ( #renaming x axis
    name = "Valence"
  ) +
  scale_y_continuous ( #renaming y axis
    name = "Danceability"
  ) +
  theme_bw(12) +
  theme(
    axis.text = element_text(size = 9, color = "black"), #resizing and recoloring axes labels
    axis.title.x = element_text(size = 11, vjust = -0.5), #resizing, shifting x axis title
    axis.title.y = element_text(size = 11, vjust = 2) #resizing, shifting y axis title
  ) +
  theme(panel.spacing.x = unit(2, "lines")) #spacing faceted plots apart
```

```
## `geom_smooth()` using formula 'y ~ x'
```



```r
#Slope estimates / Uncertainty: Setup
lm_data <- spotify_songs2 %>%
  nest(data = -c(playlist_name)) %>% #group variables by playlist_name
  mutate( #apply lm formula to each data table
    fit = map(data, ~lm(danceability ~ valence, data = .x)),
    tidy_out = map(fit, tidy) #apply tidy on each 'fit' object
  ) %>%
  unnest(cols = tidy_out) %>%
  select(-fit, -data) %>%
  filter(term!= "(Intercept)")

#Slope estimates: Visualization
lm_data %>%
  mutate( #reordering slope estimates from least to greatest
    playlist_name = fct_reorder(playlist_name, estimate)
  ) %>%
  ggplot(aes(x = estimate, y = playlist_name)) +
  stat_dist_dotsinterval( #dist parameters, setting up quantiles
    aes(dist = dist_normal(mu = estimate, sigma = std.error)),
    point_size = 4,
    fill = "skyblue",
    quantiles = 15
    ) +
  scale_x_continuous( #renaming x axis
    name = "Slope Estimate"
  ) +
  scale_y_discrete( #renaming y axis
    name = "Playlist Name"
  ) +
  theme_bw(12) +
  theme(
    axis.text = element_text(size = 9, color = "black"), #resizing and recoloring axes labels
    axis.title.x = element_text(size = 11, vjust = -0.5), #resizing, shifting x axis title
    axis.title.y = element_text(size = 11, vjust = 2) #resizing, shifting y axis title
  )
```



Discussion:

The first part of the question looks at whether or not there is a linear relationship between valence and danceability for LoFi songs. According to the linear model, we received a p-value of less that 0.05 for all three regression plots. Therefore, we can say there is a significant relationship between valence and danceability in all three plots. It is also worth noting the R-squared value for each of the three plots: In "Lofi Hip-Hop", valence only explains about 5% of the variability in danceability, meaning that there are likely many more variables responsible for variation in danceability. In "Lush Lofi" however, 20% of the variation in danceability is explained by valence, making the variable much more descriptive and valuable in this playlist. As danceability is partly measured through "rhythm stability and overall regularity" (via TidyTuesday), perhaps "Lush Lofi" contains songs that have more unpredictable components, such as fluctuating volume, for example. This might explain why Valence plays a larger role in explaining danceability here than other the other two LoFi playlists.

The second part of the question asks how these linear relationships differ among the playlists. According to the blue dots, LoFi Hip-Hop and Lo-Fi Beats have the same probabilities: 1/15 chance of being on the slope estimate, 9/15 chance of being 1 standard error away, and 5/15 chance on being 2 standard errors away. Lush Lofi has a 3/15 chance of being on the mean, 8/15 chance of being 1 standard error away, and a 4/15 chance of being 2 standard errors away. Thus, the uncertainty between our slope estimates are similar throughout the playlists. According to the black dots, Lush Lofi has the largest slope between valence and danceability, followed by Lo-Fi Beats, then Lofi Hip-Hop. One reason these slopes differ may be because, like mentioned above, other variables that explain danceability within these playlists.