Final Project - First Visualization

Group 5

Juan David Nieto Garcia — 114065428
Erdenejargal Galtsuren — 113065430

Seasonal Shifts in Music Mood: Exploring How Spotify Listening Preferences Change Across Seasons

Question: Do seasonal changes influence the emotional mood of music people listen to? Specifically, do listeners tend to play happier (higher valence) music during summer months and sadder (lower valence) music during autumn and winter?

```
file.info("merged_data.csv")$size
```

```
[1] 27104091782
```

Data Pre-Processing

Our dataset, Spotify Charts – All Audio Data (merged_data.csv), is approximately 27 GB. Loading it directly into R would exceed the memory capacity of a standard computer, so we first inspect only a sample of the data:

```
sample_data <- read_csv("merged_data.csv", n_max = 100000)
spec(sample_data)
```

```
cols(
  ...1 = col_double(),
  title = col_character(),
  rank = col_double(),
  date = col_date(format = ""),
  artist = col_character(),
  url = col_character(),
  region = col_character(),
  chart = col_character(),
  trend = col_character(),
  streams = col_double(),
  track_id = col_character(),
  album = col_character(),
  popularity = col_double(),
  duration_ms = col_double(),
```

```
  explicit = col_logical(),
  release_date = col_character(),
  available_markets = col_character(),
  af_danceability = col_double(),
  af_energy = col_double(),
  af_key = col_double(),
  af_loudness = col_double(),
  af_mode = col_double(),
  af_speechiness = col_double(),
  af_acousticness = col_double(),
  af_instrumentalness = col_double(),
  af_liveness = col_double(),
  af_valence = col_double(),
  af_tempo = col_double(),
  af_time_signature = col_double()
)
```

```
system("wc -l merged_data.csv")
```

To focus on seasonal listening trends, we removed columns that do not contribute to mood or time-based analysis (e.g., URLs, album details, and detailed audio features). These variables increase file size and memory usage without adding meaningful insight.

Because the dataset is very large (27GB+), we process it in chunks rather than loading it all at once. The cleaned dataset keeps only the essential columns such as date, streams, and af_valence.

```
# Columns to remove because they are not useful for seasonal mood analysis
cols_to_drop <- c(
  "url", "chart", "trend", "track_id", "album",
  "duration_ms", "explicit", "release_date", "available_markets",
  "af_danceability", "af_key", "af_loudness", "af_mode",
  "af_speechiness", "af_acousticness", "af_instrumentalness",
  "af_liveness", "af_tempo", "af_time_signature"
)

# Output file name
out_file <- "merged_data_trimmed.csv"
if (file.exists(out_file)) file.remove(out_file)

first_chunk <- TRUE

process_chunk <- function(chunk, pos) {
  # Drop irrelevant columns
  chunk <- chunk %>% select(-any_of(cols_to_drop))

  # Write the first chunk with headers, remaining chunks appended
  write_csv(chunk, out_file, append = !first_chunk, col_names = first_chunk)

  if (first_chunk) message("Header written.")
  message("Processed up to row: ", pos)

  first_chunk <<- FALSE
}
```

```
# Process dataset in chunks (250,000 rows at a time)
read_csv_chunked(
  "merged_data.csv",
  callback = SideEffectChunkCallback$new(process_chunk),
  chunk_size = 26174270
)
```

```
NULL
```

Check new file details

```
file.exists("merged_data_trimmed.csv")   # TRUE
```

```
[1] TRUE
```

```
file.info("merged_data_trimmed.csv")$size
```

```
[1] 2241076550
```

The trimmed file size is now approximately 2.2 GB, making it much easier to load into R. It still contains all 26 million rows, but only the relevant columns needed for seasonal analysis.

```
# Preview of the New File
read_csv("merged_data_trimmed.csv", n_max = 5)
```

```
# A tibble: 5 x 10
     X1 title         rank date       artist region streams popularity af_energy
  <dbl> <chr>        <dbl> <date>     <chr>  <chr>    <dbl>      <dbl>     <dbl>
1     0 Chantaje (f~     1 2017-01-01 Shaki~ Argen~  253019         78     0.773
2     1 Vente Pa' C~     2 2017-01-01 Ricky~ Argen~  223988         72     0.92
3     2 Reggaetón L~     3 2017-01-01 CNCO   Argen~  210943         73     0.838
4     3 Safari           4 2017-01-01 J Bal~ Argen~  173865          0     0.687
5     4 Shaky Shaky      5 2017-01-01 Daddy~ Argen~  153956          0     0.626
# i 1 more variable: af_valence <dbl>
```

```
spec(read_csv("merged_data_trimmed.csv", n_max = 0))
```

```
cols(
  X1 = col_character(),
  title = col_character(),
  rank = col_character(),
  date = col_character(),
  artist = col_character(),
  region = col_character(),
  streams = col_character(),
  popularity = col_character(),
  af_energy = col_character(),
  af_valence = col_character()
)
```

Now let's load the file

```r
all_songs <- fread("merged_data_trimmed.csv")
```

We'll start by taking a look at all of the unique countries. we are interested in doing analysis on the ones that have seasons, so some of the countries near the equator may be discarded moving forward

```r
# Get unique countries
unique_countries <- unique(all_songs$region)

# Preview
head(unique_countries, 20)  # first 20 countries
```

```
 [1] "Argentina"          "Australia"          "Brazil"
 [4] "Austria"            "Belgium"            "Colombia"
 [7] "Bolivia"            "Denmark"            "Bulgaria"
[10] "Canada"             "Chile"              "Costa Rica"
[13] "Czech Republic"     "Finland"            "Dominican Republic"
[16] "Ecuador"            "El Salvador"        "Estonia"
[19] "France"             "Germany"
```

```r
length(unique_countries)    # total number of unique countries
```

```
[1] 70
```

Now we'll group all of the songs by region and check which countries have the most data:

```r
country_counts <- all_songs %>%
  group_by(region) %>%
  summarise(song_count = n()) %>%
  arrange(desc(song_count))

# Preview top 10 countries
head(country_counts, 67)
```

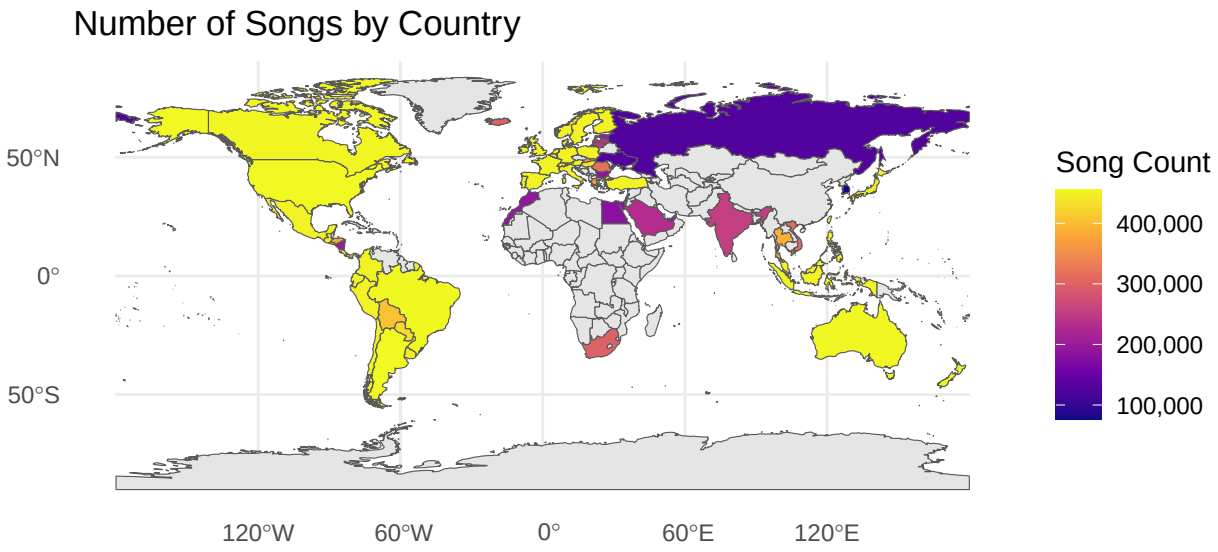
```
# A tibble: 67 x 2
   region         song_count
   <chr>               <int>
 1 Argentina          455311
 2 United States      455085
 3 Austria            454596
 4 Brazil             454441
 5 Australia          453117
 6 Canada             452296
 7 Global             451804
 8 United Kingdom     450732
 9 Switzerland        449658
10 Malaysia           449354
# i 57 more rows
```

```r
# had to change the name of the US since it would not appear in the plot
region_summary <- all_songs %>%
```

```r
  group_by(region) %>%
  summarise(song_count = n()) %>%
  mutate(
    region = case_when(
      region == "United States" ~ "United States of America",
      TRUE ~ region
    )
  )
# Load world map
world <- ne_countries(scale = "medium", returnclass = "sf")

# Join map with your data (region -> name)
map_data <- left_join(world, region_summary, by = c("name" = "region"))

# Plot: color = number of songs
ggplot(map_data) +
  geom_sf(aes(fill = song_count)) +
  scale_fill_viridis_c(
    option = "plasma",
    na.value = "grey90",
    labels = scales::comma
  ) +
  theme_minimal() +
  labs(
    title = "Number of Songs by Country",
    fill = "Song Count"
  )
```




```
nrow(region_summary)
```

```
[1] 70
```

```
region_summary$region
```

```
 [1] "Andorra"                "Argentina"
 [3] "Australia"              "Austria"
 [5] "Belgium"                "Bolivia"
 [7] "Brazil"                 "Bulgaria"
 [9] "Canada"                 "Chile"
[11] "Colombia"               "Costa Rica"
[13] "Czech Republic"         "Denmark"
[15] "Dominican Republic"     "Ecuador"
[17] "Egypt"                  "El Salvador"
[19] "Estonia"                "Finland"
[21] "France"                 "Germany"
[23] "Global"                 "Greece"
[25] "Guatemala"              "Honduras"
[27] "Hong Kong"              "Hungary"
[29] "Iceland"                "India"
[31] "Indonesia"              "Ireland"
[33] "Israel"                 "Italy"
[35] "Japan"                  "Latvia"
[37] "Lithuania"              "Luxembourg"
```

```
[39] "Malaysia"                "Mexico"
[41] "Morocco"                 "Netherlands"
[43] "New Zealand"             "Nicaragua"
[45] "Norway"                  "Panama"
[47] "Paraguay"                "Peru"
[49] "Philippines"             "Poland"
[51] "Portugal"                "Romania"
[53] "Russia"                  "Saudi Arabia"
[55] "Singapore"               "Slovakia"
[57] "South Africa"            "South Korea"
[59] "Spain"                   "Sweden"
[61] "Switzerland"             "Taiwan"
[63] "Thailand"                "Turkey"
[65] "Ukraine"                 "United Arab Emirates"
[67] "United Kingdom"          "United States of America"
[69] "Uruguay"                 "Vietnam"
```

We are going to remove countries from latin america, some from the middle east and india since these don't have regular seasons due to their proximity to the equator

```
latin_america <- c(
  "Argentina", "Bolivia", "Brazil", "Chile", "Colombia", "Costa Rica",
  "Dominican Republic", "Ecuador", "El Salvador", "Guatemala", "Honduras",
  "Mexico", "Nicaragua", "Panama", "Paraguay", "Peru", "Uruguay"
)

middle_east <- c(
  "Israel", "Turkey", "Saudi Arabia", "United Arab Emirates", "Egypt"
)

exclude <- c(latin_america, middle_east, "India")
```

```
# Filter out excluded regions from all_songs
all_songs_filtered <- all_songs %>%
  filter(!region %in% exclude)

# Check result
nrow(all_songs_filtered)   # number of remaining rows
```

```
[1] 17387946
```

```
length(unique(all_songs_filtered$region))  # number of remaining countries
```

```
[1] 47
```

Now we'll make a new csv file with the filtered data and remove the data from `all_songs` to avoid issues with memory:

```
# Write the filtered data to a new CSV file
fwrite(all_songs_filtered, "merged_data_filtered.csv")

# Remove the large original object to free memory
rm(all_songs)
```

We group the songs by region and filter for only canada to have a preliminary view on the data

```
# Group by country (region)
all_songs_filtered <- all_songs_filtered %>%
  group_by(region)

# Extract only the songs for Canada
Canada_songs <- all_songs_filtered %>%
  filter(region == "Canada")

# Optional: check how many songs Canada has
nrow(Canada_songs)
```

```
[1] 452296
```

```
head(Canada_songs)
```

```
# A tibble: 6 x 10
# Groups:   region [1]
     X1 title         rank date       artist region streams popularity af_energy
  <int> <chr>        <int> <IDate>    <chr>  <chr>    <int>      <int>     <dbl>
1  1071 Starboy          1 2017-01-01 The W~ Canada  139175         23     0.594
2  1072 Closer           2 2017-01-01 The C~ Canada  128481         86     0.524
3  1073 Fake Love        3 2017-01-01 Drake  Canada  118532         73     0.481
4  1074 Bad and Bou~     4 2017-01-01 Migos  Canada  100524          1     0.666
5  1075 One Dance        5 2017-01-01 Drake~ Canada   90257         23     0.619
6  1076 Black Beatl~     6 2017-01-01 Rae S~ Canada   90233         75     0.632
# i 1 more variable: af_valence <dbl>
```

Check for date ranges on the subset

```
# Get min and max dates for Canada
Canada_date_range <- Canada_songs %>%
  summarise(
    min_date = min(date, na.rm = TRUE),
    max_date = max(date, na.rm = TRUE)
  )

Canada_date_range
```

```
# A tibble: 1 x 3
  region min_date   max_date
  <chr>  <IDate>    <IDate>
1 Canada 2017-01-01 2021-12-31
```

Now let's take a closer look at the songs from 2017

```
# Filter Canada songs for 2017 using IDate comparison
Canada_songs_2017 <- all_songs_filtered %>%
  filter(
    region == "Canada" &
```

```
    date >= as.IDate("2017-01-01") &
    date <= as.IDate("2017-12-31")
  )

# Check result
nrow(Canada_songs_2017)
```

```
[1] 90555
```

```
tail(Canada_songs_2017)
```

```
# A tibble: 6 x 10
# Groups:   region [1]
        X1 title      rank date       artist region streams popularity af_energy
     <int> <chr>     <int> <IDate>    <chr>  <chr>    <int>      <int>     <dbl>
1 25987714 Despacito    45 2017-05-31 Despa~ Canada      NA         13     0.406
2 25987715 Put The ~    46 2017-05-31 Goldf~ Canada      NA          0     0.916
3 25987716 What You~    47 2017-05-31 Withi~ Canada      NA          0     0.881
4 25987717 Unforget~    48 2017-05-31 JLuv ~ Canada      NA         NA    NA
5 25987718 Adeline      49 2017-05-31 alt-J  Canada      NA         10     0.329
6 25987719 Rollin (~    50 2017-05-31 Calvi~ Canada      NA         57     0.762
# i 1 more variable: af_valence <dbl>
```

sort by date

```
# Sort by date ascending
Canada_songs_2017 <- Canada_songs_2017 %>%
  arrange(date)

# Quick check
head(Canada_songs_2017)
```

```
# A tibble: 6 x 10
# Groups:   region [1]
     X1 title         rank date       artist region streams popularity af_energy
  <int> <chr>        <int> <IDate>    <chr>  <chr>    <int>      <int>     <dbl>
1  1071 Starboy          1 2017-01-01 The W~ Canada  139175         23     0.594
2  1072 Closer           2 2017-01-01 The C~ Canada  128481         86     0.524
3  1073 Fake Love        3 2017-01-01 Drake  Canada  118532         73     0.481
4  1074 Bad and Bou~     4 2017-01-01 Migos  Canada  100524          1     0.666
5  1075 One Dance        5 2017-01-01 Drake~ Canada   90257         23     0.619
6  1076 Black Beatl~     6 2017-01-01 Rae S~ Canada   90233         75     0.632
# i 1 more variable: af_valence <dbl>
```

```
tail(Canada_songs_2017)
```

```
# A tibble: 6 x 10
# Groups:   region [1]
        X1 title      rank date       artist region streams popularity af_energy
     <int> <chr>     <int> <IDate>    <chr>  <chr>    <int>      <int>     <dbl>
1 24760710 "LOYALTY~   195 2017-12-31 Kendr~ Canada   22926         78     0.535
```

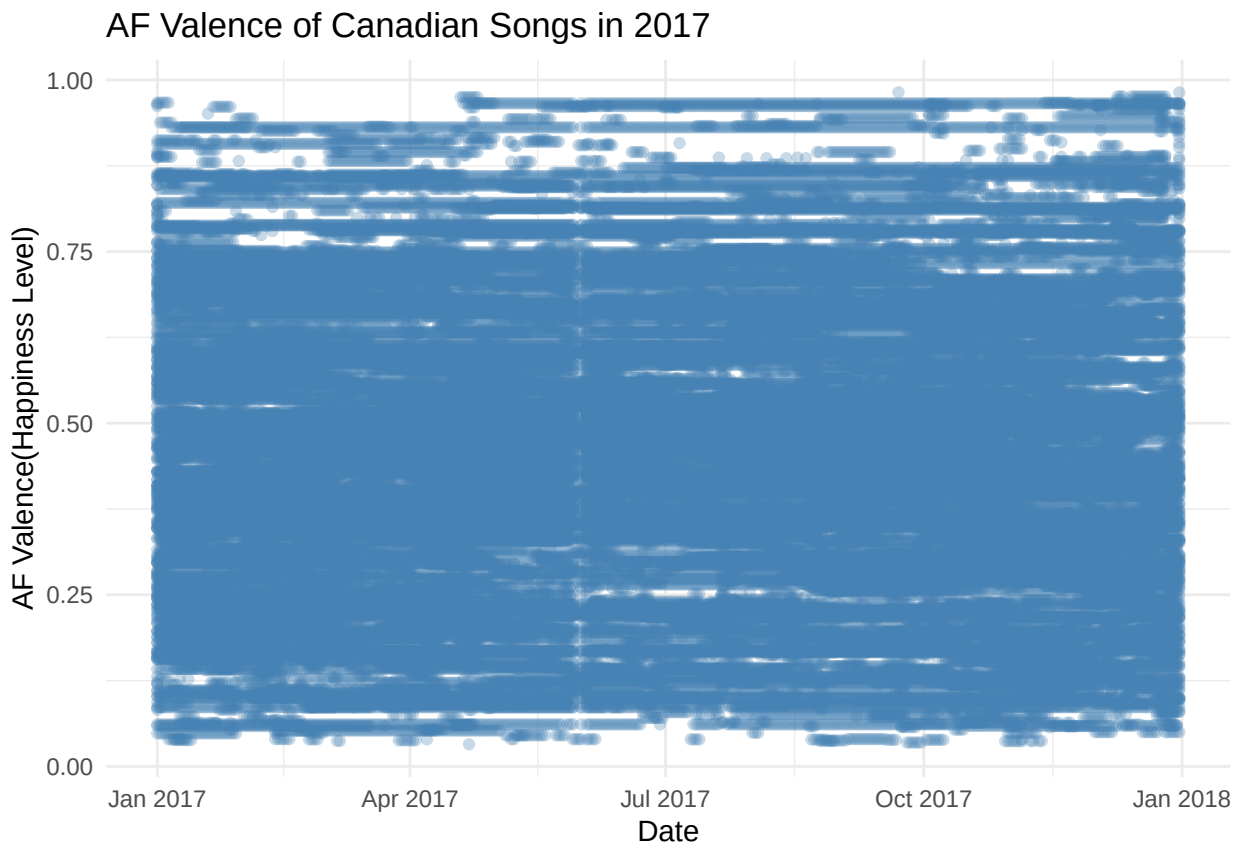
```
2 24760711 "Happy -~   196 2017-12-31 Pharr~ Canada   22899          0     0.822
3 24760712 "Send Me~   197 2017-12-31 Ruste~ Canada   22608         72     0.704
4 24760713 "Beat It"   198 2017-12-31 Micha~ Canada   22505         76     0.867
5 24760714 "Corazón"   199 2017-12-31 Malum~ Canada   22279          0     0.768
6 24760715 "Come On~   200 2017-12-31 Dexys~ Canada   22209          0     0.658
# i 1 more variable: af_valence <dbl>
```

First plot - Understanding AF Valence (Song Happiness Level)

`af_valence` is a metric from Spotify's audio features that measures the **musical positivity or happiness** of a song.
- Values close to **1** → happy, cheerful, euphoric songs
- Values close to **0** → sad, depressed, angry songs

To explore how overall music mood changes across time, we first plot raw valence values for Canadian songs in 2017.

```
ggplot(Canada_songs_2017, aes(x = date, y = af_valence)) +
  geom_point(alpha = 0.3, color = "steelblue", na.rm = TRUE) +
  theme_minimal() +
  labs(
    title = "AF Valence of Canadian Songs in 2017",
    x = "Date",
    y = "AF Valence(Happiness Level)"
  )
```



The initial scatter plot shows individual song valence values across time, but the data appears highly scattered and noisy, making it difficult to identify any meaningful trends.

To better understand overall mood patterns, we calculate the weighted daily average of af_valence. In this method, each song's valence is weighted by its number of streams. This way, songs that were listened to more frequently have a greater impact on the daily mood score.

We chose this approach because streaming counts reflect how many people are listening to a song and, therefore, give a more accurate representation of the collective music mood among listeners on each day.

Weighted Daily Valence Calculation

To understand how music mood (valence) changes across the year, we calculate the **weighted average valence per day**.
This means songs with more streams contribute more to the daily valence score — making the result more representative of what people were actually listening to.

```
Canada_daily_valence <- Canada_songs_2017 %>%
  filter(!is.na(af_valence) & !is.na(streams)) %>%
  group_by(date) %>%
  summarise(
    weighted_valence = sum(af_valence * streams) / sum(streams)
  ) %>%
  ungroup()

# Quick preview of the results
head(Canada_daily_valence)
```

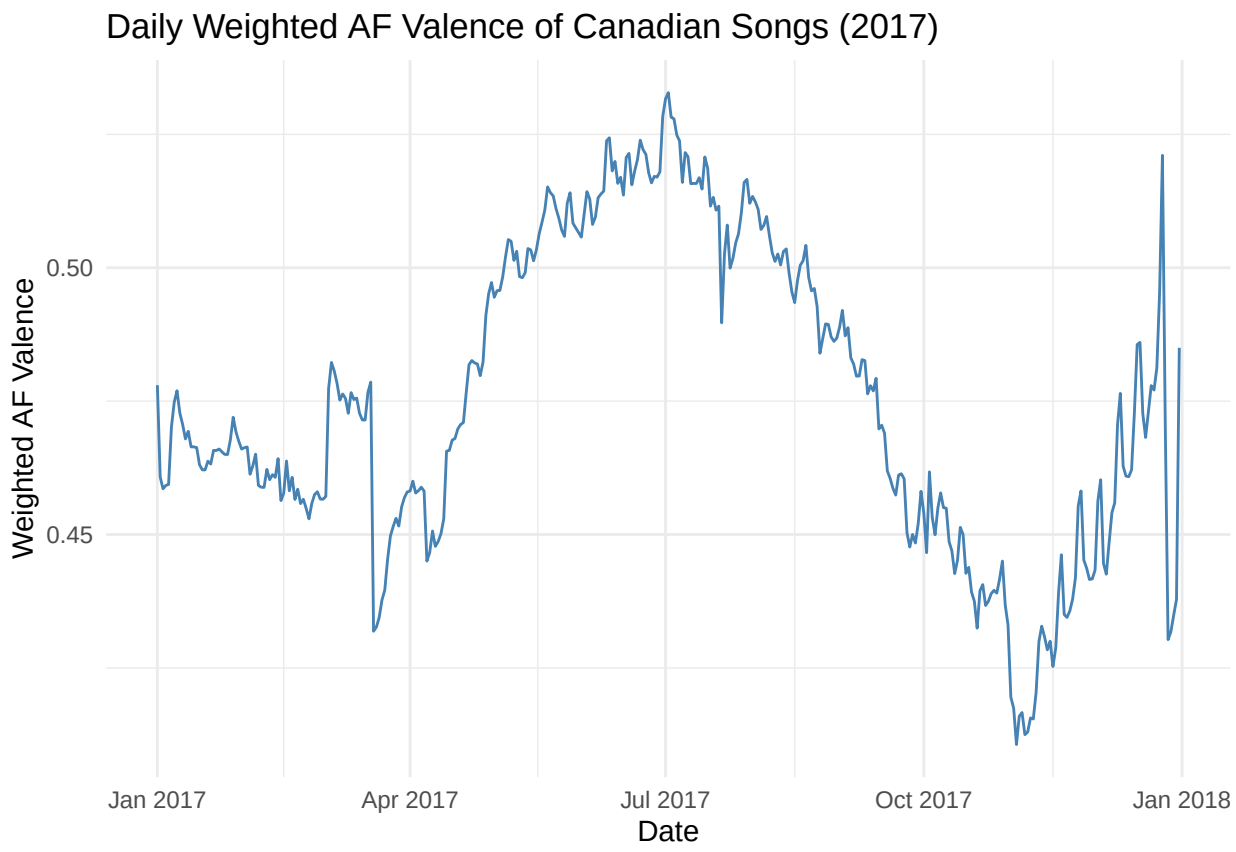
```
# A tibble: 6 x 2
  date       weighted_valence
  <IDate>               <dbl>
1 2017-01-01            0.478
2 2017-01-02            0.461
3 2017-01-03            0.459
4 2017-01-04            0.459
5 2017-01-05            0.459
6 2017-01-06            0.470
```

- af_valence * streams gives each song's valence contribution proportional to its streams.
- sum(af_valence * streams) / sum(streams) is the weighted mean for that day.
- group_by(date) ensures one row per day.
- Canada_daily_valence is now a tidy daily vector of weighted valence, ready to plot.

```
Sys.setlocale("LC_TIME", "C")  # "C" is standard POSIX locale (English)
```

```
[1] "C"
```

```r
ggplot(Canada_daily_valence, aes(x = date, y = weighted_valence)) +
  geom_line(color = "steelblue") +
  theme_minimal() +
  labs(
    title = "Daily Weighted AF Valence of Canadian Songs (2017)",
    x = "Date",
    y = "Weighted AF Valence"
  )
```



Conclusion

Based on the weighted daily AF valence of Canadian Spotify streams in 2017, we can observe a clear seasonal trend in music mood preferences.

- Valence levels rise from early spring and peak around **July (summer)**, indicating that listeners tend to prefer **happier, more positive songs** during this period.
- After summer, valence levels begin to decline, reaching lower values during **autumn**, suggesting a shift toward **sadder or moodier music**.

These results support our research question — that seasonal changes may influence the emotional tone of the music people listen to, with summer associated with happier music and autumn showing a preference for lower-valence songs.