1. In Lab 3, you wrangled data from Essentia, Essentia models and LIWC. Rework your solution to Lab 3 using `tidyverse` (Wickham et al., 2019) instead of base `R`. Specifically, rewrite your code for steps 1-4 of task 2 using `tidyverse` (Wickham et al., 2019). Make sure to address any issues I noted in your code file, and ensure that your code runs in the directory as it is set up.

```
################################################################################
# 2: Lab Coding Task: Compile Data from Essentia
################################################################################

library("jsonlite")
help("jsonlite")
library("tidyverse")
help("tidyverse")

# Step 1

# File Name + Extraction
current.filename <- ("The Front Bottoms-Talon Of The Hawk-Au Revoir (Adios).json")
curr.file.parts <- str_split(current.filename, "-", simplify = TRUE)
curr.artist.file <- curr.file.parts[1]
curr.album.file <- curr.file.parts[2]
curr.track.file <- curr.file.parts[3] |>
  str_remove(".json$")

# Load JSON file
json.storing <- fromJSON(file.path("EssentiaOutput", current.filename))

# Extract metadata
overall.loudness <- json.storing$lowlevel$loudness_ebu128$integrated
spectral.energy <- json.storing$lowlevel$spectral_energy
dissonance <- json.storing$lowlevel$dissonance
pitch.salience <- json.storing$lowlevel$pitch_salience
bpm <- json.storing$rhythm$bpm
beats.loudness <- json.storing$rhythm$beats_loudness
danceability <- json.storing$rhythm$danceability
tuning.frequency <- json.storing$tonal$tuning_frequency


######## Step 2 #########
help("map_dfr") # cite purrr package for map_dfr
                # documentation says purrr:map functions
                # can be thought of as a replacement to for loops

data.extraction <- function(json.file.names){
    df <- map_dfr(json.file.names, function(file){
      file.parts <- str_split(file, "-", simplify = TRUE)
      artist <- file.parts[1]
      album <- file.parts[2]
      track <- file.parts[3]|>
        str_remove(".json$")

    json.storage <- fromJSON(file.path("EssentiaOutput", current.filename))

    tibble(
      artist = artist,
      album = album,
      track = track,
      overall_loudness = json.storage$lowlevel$loudness_ebu128$integrated,
      spectral_energy = json.storage$lowlevel$spectral_energy,
      dissonance = json.storage$lowlevel$dissonance,
      pitch_salience = json.storage$lowlevel$pitch_salience,
      bpm = json.storage$rhythm$bpm,
      beats_loudness = json.storage$rhythm$beats_loudness,
      danceability = json.storage$rhythm$danceability,
      tuning_frequency = json.storage$tonal$tuning_frequency
    )

  })
  return(df)
}

# List of .json files
json.file.names <- list.files("EssentiaOutput", pattern = "\\.json$")
df <- data.extraction(json.file.names)
view(df)

################################################################################
# Step 3: Essentia Model Cleanup
################################################################################
```

```r
# Load csv file
essentiamodeldf <- read_csv(file.path("EssentiaOutput","EssentiaModelOutput.csv"))
#View(essentiamodeldf)
dim(essentiamodeldf)
```

```
## [1] 181  49
```

```r
cleaned.essentia.model <- essentiamodeldf |>
  mutate(
    # Create Two New Columns of means of 3 provided rows
    valence = rowMeans(essentiamodeldf[,c("deam_valence", "emo_valence", "muse_valence")], na.rm = T),
    arousal = rowMeans(essentiamodeldf[,c("deam_arousal", "emo_arousal", "muse_arousal")], na.rm = T),
    # Discogs-EffNet and MSD-MusiCNN Extractors
    aggressive = rowMeans(essentiamodeldf[,c("eff_aggressive", "nn_aggressive")], na.rm = T),
    happy = rowMeans(essentiamodeldf[,c("eff_happy", "nn_happy")], na.rm = T),
    party = rowMeans(essentiamodeldf[,c("eff_party", "nn_party")], na.rm = T),
    relax = rowMeans(essentiamodeldf[,c("eff_relax", "nn_relax")], na.rm = T),
    sad = rowMeans(essentiamodeldf[,c("eff_sad", "nn_sad")], na.rm = T),
    acoustic = rowMeans(essentiamodeldf[,c("eff_acoustic", "nn_acoustic")], na.rm = T),
    electronic = rowMeans(essentiamodeldf[,c("eff_electronic", "nn_electronic")], na.rm = T),
    instrumental = rowMeans(essentiamodeldf[,c("eff_instrumental", "nn_instrumental")], na.rm = T)
  ) |>
  rename(timbreBright = eff_timbre_bright) |>
  # Removed all nonessential columns
  select("artist",
         "album",
         "track",
         "valence",
         "arousal",
         "aggressive",
         "happy",
         "party",
         "relax",
         "sad",
         "acoustic",
         "electronic",
         "instrumental",
         "timbreBright")

view(cleaned.essentia.model)

################################################################################
# Step 4: LIWC Merge
################################################################################

# Load csv file
LIWCOutputdf <- read_csv(file.path("LIWCOutput", "LIWCOutput.csv"))

final.merge <- df |>
  left_join(cleaned.essentia.model)|>
  left_join(LIWCOutputdf)|>
  rename(funct = "function")

dim(final.merge) # matches the original lab 3 dimensions
```

```
## [1] 181 140
```

To change my lab 3 to use the `tidyverse` package (Wickham et al., 2019), I started with step 1 by creating a function that made it easier to input all of the `.json` files. Functions are good to use as they can be tested instead of running it more manually to make sure there are no errors. I thought this was a good way to incorporate the use of functions as practice as well as practicality. I also found the function `map_dfr()` which is part of the `purr` package (Wickham and Henry, 2025) within `tidyverse` (Wickham et al., 2019). The documentation from the `tidyverse` (Wickham et al., 2019) website said that purrr:map functions can be thought of as a replacement to for loops. The `map_dfr()` returns data frames (or tibbles) by row-binding the outputs together. It is designed to iterate over elements which is a sufficient replacement for the `for()` loop in the previous lab with base `R`. I also used `str_remove()` (Wickham, 2023) which removes the `.json` at the end of the file name. I then created a tibble for each `.json` file instead of creating an empty data frame. The `map_dfr()` binds the tibble together to create the neat final tibble. For step 3, I used the function `mutate()` which allows for the code to be readable and concise. The pipe throughout this step allows for me to not have to redefine what I am manipulating since I am changing multiple parts of one tibble. Lastly for step 4, I used `left_join()`. This automatically removes the repeated columns in the different tibbles I am combining. I cross checked the final dimensions of my final merged tibble and the one from last lab finding that they were equivalent: 181 rows and 140 columns.

References

Wickham, H. (2023). *stringr: Simple, Consistent Wrappers for Common String Operations.* R package version 1.5.1.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., and Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686.

Wickham, H. and Henry, L. (2025). *purrr: Functional Programming Tools.* R package version 1.0.4.