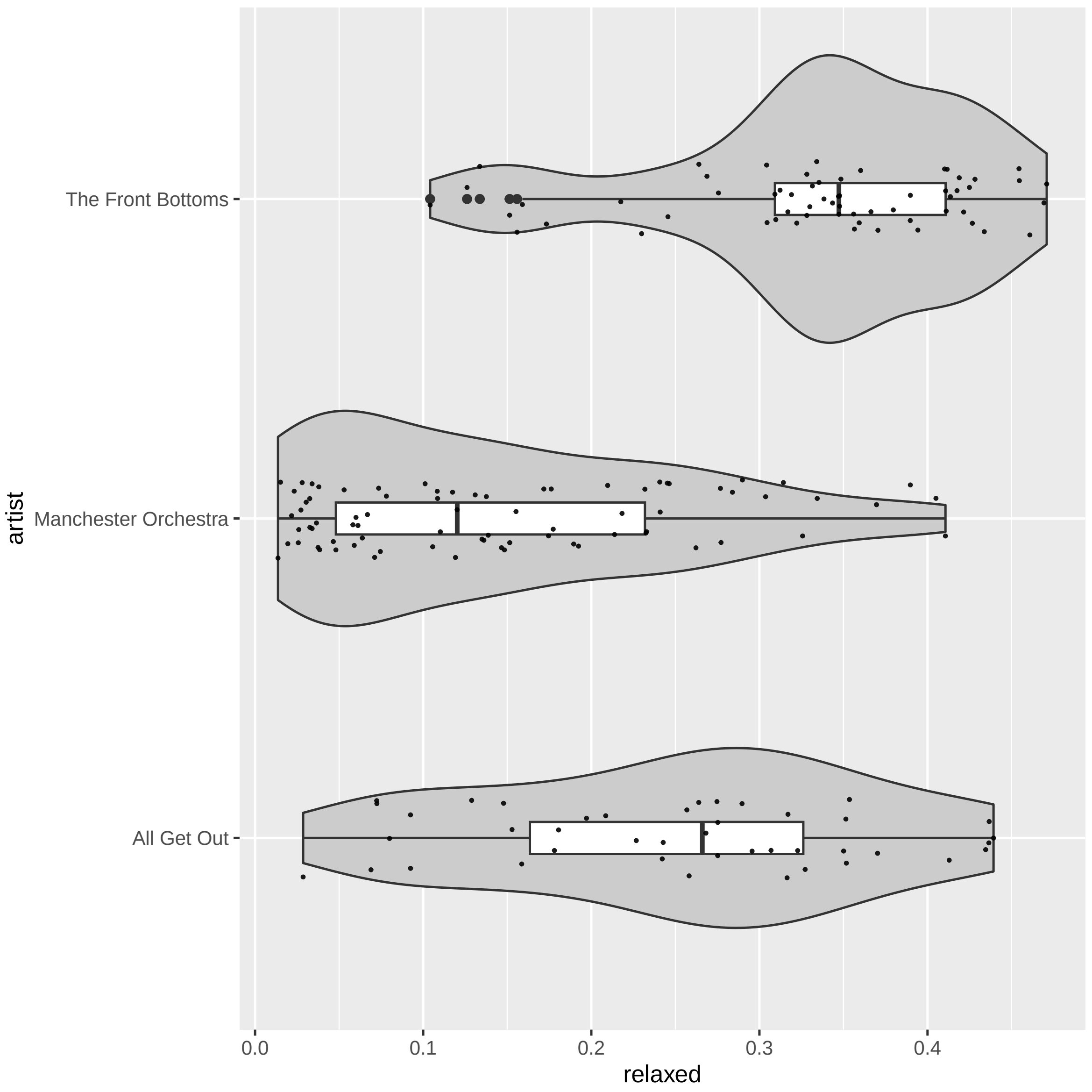
1. In Lab 3, you wrangled data from Essentia, Essentia models and LIWC. Rework your solution to Lab 3 using `tidyverse` (Wickham et al., 2019) instead of base `R`. Specifically, rewrite your code for steps 1-4 of task 2 using `tidyverse` (Wickham et al., 2019). Make sure to address any issues I noted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

```r
# Henry Sun
# Homework 5
# 2/26/25


# Step 0
library("stringr")
library("jsonlite")
library(tidyverse)


#Step 1, Front Bottoms Example
current.filename = "The Front Bottoms-Talon of the Hawk-Au Revoir (Adios).json"
currentfile.split = str_split_1(current.filename, "-")

# artist, album, track
artist = currentfile.split[1]
album = currentfile.split[2]
track = str_sub(currentfile.split[3], start = 0, end = -6)

# Essentia Output
curr.json = (fromJSON(paste("EssentiaOutput/", current.filename, sep="")))

# lowlevel
overall.loudness = curr.json$lowlevel$loudness_ebu128$integrated
spectral.energy = curr.json$lowlevel$spectral_energy$mean
dissonance = curr.json$lowlevel$dissonance$mean
pitch.salience = curr.json$lowlevel$pitch_salience$mean

# rhythm
bpm = curr.json$rhythm$bpm
beats.loudness = curr.json$rhythm$beats_loudness$mean
danceability = curr.json$rhythm$danceability

# tonal
tuning.freq = curr.json$tonal$tuning_frequency

# data from example file
currfile.data = tibble(overall.loudness, spectral.energy, dissonance, pitch.salience, bpm, beats.loudness,
                  danceability, tuning.freq)


# Step 2, repeat for all .json files
# load files
all.files = (list.files("EssentiaOutput", recursive=TRUE))

# only check files with .json
json.check = str_count(all.files, pattern=".json")
all.json = all.files[which(json.check == 1)]

# create tibble
json.data = tibble(
  artist = character(),
  album = character(),
  overall.loudness = numeric(),
  spectral.energy = numeric(),
  dissonance = numeric(),
  pitch.salience = numeric(),
  bpm = numeric(),
  beats.loudness = numeric(),
  danceability = numeric(),
  tuning.freq = numeric()
)

# repeat same for all files in step 1
for (i in 1:length(all.json)){
  curr.file = fromJSON(paste("EssentiaOutput/", all.json[i], sep = ""))
  currfile.split =  str_split_1(all.json[i], "-")

  artist = currfile.split[1]
  album = currfile.split[2]
  track = str_sub(currfile.split[3], start = 0, end = -6)

  overall.loudness = curr.file$lowlevel$loudness_ebu128$integrated
  spectral.energy = curr.file$lowlevel$spectral_energy$mean
  dissonance = curr.file$lowlevel$dissonance$mean
  pitch.salience = curr.file$lowlevel$pitch_salience$mean
  bpm = curr.file$rhythm$bpm
  beats.loudness = curr.file$rhythm$beats_loudness$mean
  danceability = curr.file$rhythm$danceability
  tuning.freq = curr.file$tonal$tuning_frequency

  # insert into tibble
  # create row of data for each song, add to tibble
  json.data = bind_rows(json.data, tibble(
    artist, album, track, overall.loudness,
    spectral.energy, dissonance, pitch.salience, bpm, beats.loudness,
    danceability, tuning.freq))
}


# Step 3
# read csv into a tibble
essentia.file = read_csv("EssentiaOutput/EssentiaModelOutput.csv")
# create new cols using mutate,
# use rowMeans() + cbind to find means for some features
# I was trying to use bind_cols but it displayed a bunch of messages when running
essentia.data <- essentia.file |>
  mutate(valence = rowMeans(cbind(deam_valence, emo_valence, muse_valence)),
         arousal = rowMeans(cbind(deam_arousal, emo_arousal, muse_arousal)),
         aggressive = rowMeans(cbind(eff_aggressive, nn_aggressive)),
         happy = rowMeans(cbind(eff_happy, nn_happy)),
         party = rowMeans(cbind(eff_party, nn_party)),
         relaxed = rowMeans(cbind(eff_relax, nn_relax)),
         sad = rowMeans(cbind(eff_sad, nn_sad)),
         acoustic = rowMeans(cbind(eff_acoustic, nn_acoustic)),
         electric = rowMeans(cbind(eff_electronic, nn_electronic)),
         instrumental = rowMeans(cbind(eff_instrumental, nn_instrumental))) |>
# rename timbreBright
  rename(timbreBright = eff_timbre_bright) |>
# select features
  select(artist, album, track, valence, arousal, aggressive, happy, party, relaxed,
         sad, acoustic, electric, instrumental, timbreBright)


# Step 4
# join all data into one file, grouping by artist, album, track (eliminate dupes)
liwc.data = read_csv("LIWCOutput/LIWCOutput.csv")
all.data <- essentia.data |>
  left_join(json.data, by = c("artist", "album", "track")) |>
  left_join(liwc.data, by = c("artist", "album", "track")) |>
# rename function
  rename("funct" = "function")


#Step 5
training.data = filter(all.data, track != "Allentown")
testing.data = filter(all.data, track == "Allentown")

training.csv = write_csv(x=training.data, "trainingdata.csv")
testing.csv = write_csv(x=testing.data, "testingdata.csv")


# Coding challenge, make a graph
# violin plot for relaxed values
relaxed.plot <- ggplot(read_csv("trainingdata.csv"), aes(x=artist, y=relaxed))+
  geom_violin(fill="grey80")+
  geom_boxplot(width = 0.1)+
  geom_jitter(color = "black", size = 0.4, alpha = 0.9, width = 0.125)+
  xlab("artist")+
  ylab("relaxed")+
  coord_flip()
relaxed.plot
```



References

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.