STA303/1002: Mixed assessment (untimed component)

Class playlist

DJ Jiahao(Green) Bai; ID: 1005804097

Contents

Assessment details

Name: Mixed assessment
Type (Main, Mini or Basket): Main
Value: 20% (Path A) or 25% (Path B)
Due: Wednesday, March 16, 2022; assessment window from 8:00 a.m. ET to 8:00 p.m. ET
Submission instructions: Timed component via Quercus quizzes (2 x 50 minutes, 1 attempt each, no pausing within each quiz) and MarkUs (10 percentage point penalty for not submitting required files)
Accommodations and extension policy: In the case of a personal illness/emergency, a declaration can be made, but must be submitted no more than 3 days after the due date. Extensions may be requested through the same form up to 48 hours before the due date.

The mixed assessment has THREE components:

- Untimed guided analysis (this), which must be submitted (both Rmd AND PDF) on MarkUs.
- There is only ONE untimed component, it is relevant to both timed parts.
- Timed assessment (PART A) (50 minutes; 12-hour assessment window is 8:00 a.m. to 8:00 p.m. ET Wednesday, March 16).
- Timed assessment (PART B) (50 minutes; 12-hour assessment window is 8:00 a.m. to 8:00 p.m. ET Wednesday, March 16).

The timed components are split across two timed quizzes so you can take a break in the middle if you wish. I.e., you could do them one after the other, or plan to do one, take an hour break and do the next one. Whatever suits you is good, as long as you submit everything before 8:00 p.m. ET.

How your grade is calculated

- The 98% of your mixed grade is based on your performance on the **timed** component.

- 2% of your grade is based on the correctness of your Rmd. There will be a student facing autotest you can run on your submissions to check if the objects required are there and appear to be mostly sensible (note, this doesn't guarantee in all cases that they are fully RIGHT, just that they passes the checks set up for this component).

- If you do the timed components, but DON'T submit the appropriate Rmd and PDF to MarkUs by the end of the window, there is an **additional** 10 percentage point penalty.
  - Note the file name requirements: `sta303-w22-mixed.Rmd` and `sta303-w22-mixed.pdf`.
  - You can upload as many times as you like before the end of the window, so make sure you upload a 'safety' copy of your Rmd and PDF once you have started working on it.

Instructions

Before making any changes in this Rmd, you should Knit it to make sure it works. I would recommend renaming this PDF to something like 'original_instructions.pdf', so you have a copy that won't be overwritten.

1. Update the `yaml` at the top of this document to have your name and your student ID. There are TWO places you need to do this for each one, probably on lines 4 and 12. I.e., replace the square brackets and everything inside them with the appropriate details. Your student ID is all numbers (usually 10, sometimes 8 or 9), it is NOT your UTORid.
2. Complete the guided analysis below. You will want to complete this BEFORE attempting your timed assessment.
3. Complete your timed assessments (two parts). THey will require your work in this document, as well general STA303 content knowledge.
4. Knit this .Rmd to .pdf and submit BOTH files to the correct dropbox on MarkUs.

Setting up your libraries

If you are working on this on the Jupyter Hub, the `tidyverse`, `lmtest`, `rvest` and `polite` packages will already be installed.

Note: **Do not add any additional libraries/packages**. You do not need them to complete these tasks and they may interfere with the autochecking of your submission.

```
# Working locally? RUN THIS CHUNK FIRST!
  # You should only need to run it once on your local machine.
# On the JupyterHub, you may need to run it at the beginning of each new session.
# These all SHOULD be installed though.

# These are the packages you will need for this activity.
packages_needed <- c("tidyverse", "lmtest",
                     "rvest", "polite")

package.check <- lapply(
  packages_needed,
  FUN = function(x) {
    if (!require(x, character.only = TRUE)) {
      install.packages(x, dependencies = TRUE,
      repos = "https://cloud.r-project.org/")}})

# Remove objects no longer needed
rm(packages_needed, package.check)
```

```
# Run libraries for easy access to the functions we'll be using
library(tidyverse)
library(lmtest)
library(polite)
library(rvest)
```

Spotify data

Spotify is a streaming music service. It provides an API for accessing its data and `spotifyr`[Charlie Thompson, Josiah Parry, Donal Phipps and Tom Wolff (2020).

> spotifyr: R Wrapper for the 'Spotify' Web API. R package version 2.1.1. http://github.com/charlie86/spotifyr] is an R package that helps facilitate this.

In return for buying her pizza, my sister went through all your (anonymized) song recommendations from the Getting to Know You Survey and the Module 3 check-in into this Spotify playlist: https://open.spotify.com/playlist/7wnSMzB7ZrFBHODG0ycJIV

You DO NOT need a Spotify account to be able to do this assessment.

variable	class	description
track_id	character	Unique ID for the track
track_name	character	Track name
artist	character	Track artist(s) (multiple credited artists separated by semicolons
in_library	numeric	1 if the track was in my (Prof. B's) library before being recommended by students this semester
track_popularity	double	Popularity score (0-100) based on track play and recency. Higher is better
explicit	logical	TRUE or FALSE for if the songs is rated 'explicit'
danceability	double	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
energy	double	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.
loudness	double	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.
mode	double	Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
speechiness	double	Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.

variable	class	description
acousticness	double	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
instrumentalness	double	Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
liveness	double	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
valence	double	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).
tempo	double	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.

Part 1: Exploring the Spotify data

1A) Load and look

1. Load the `class_playlist.csv` data from the `data` folder. Save it as `playlist_data`
2. Run `glimpse` on this data. (You may also like to click on it in the Environment pane to view it, or run `View(playlist_data)` in the console, but don't add it to a chunk.)

```
# Load data
playlist_data <- read_csv("data/class_playlist.csv")
```

```
## Rows: 557 Columns: 16
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): track_id, track_name, artist
## dbl (12): in_library, track_popularity, danceability, energy, loudness, mode...
## lgl  (1): explicit
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

1B) Vizualize track features

Explore the audio features of the songs in our dataset in ONE faceted image.

i. Create a tibble called `long_data` from `playlist_data`, remove `track_name` and `artist`, and change the dataset to be three columns: `track_id`, `name` and `value`.

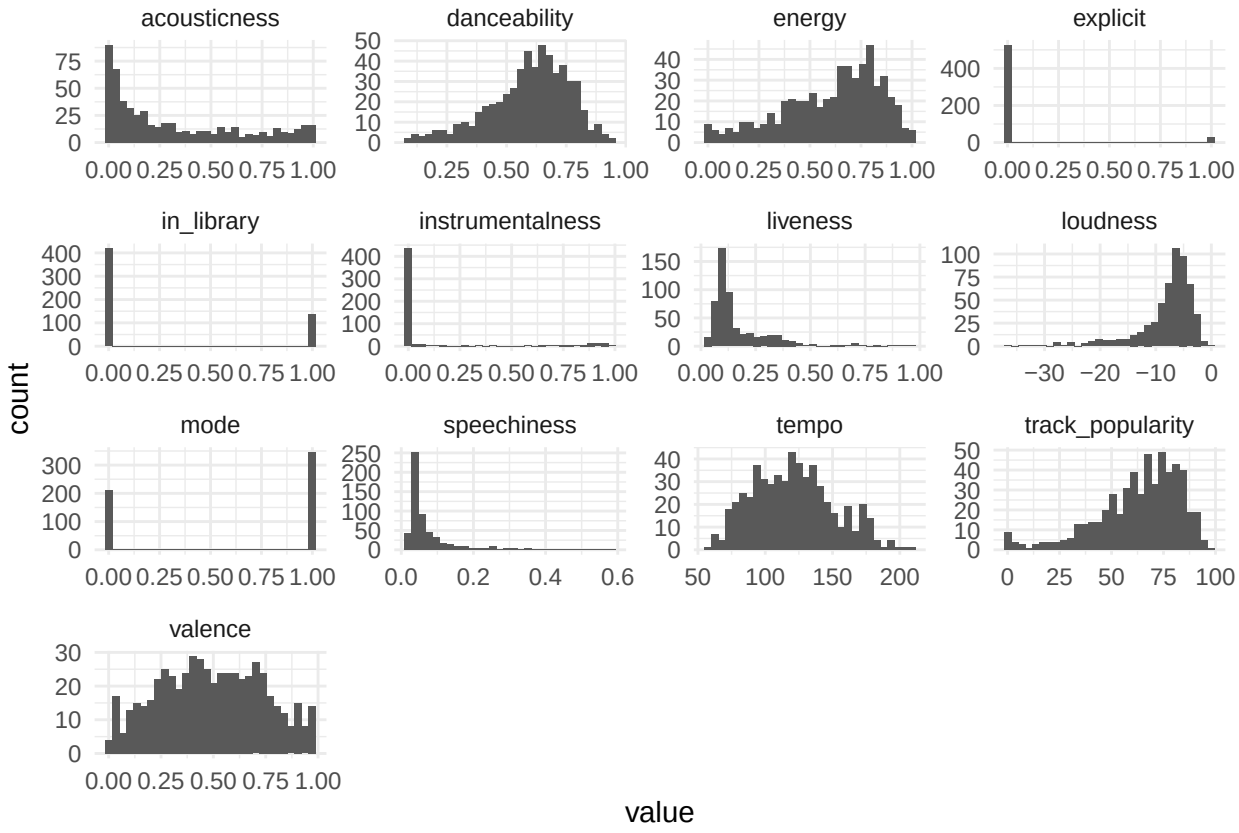
ii. Use `long_data` to create a ggplot image that shows the distributions of each of the variables in the datasets (not including `track_id`). Histogram with `bins = 30` is appropriate.

Use the following template code:

```
long_data %>%
  ________ +
  ________ +
  facet_wrap(~______, scales = "free") +
  theme_minimal()
```

```
# Create the tibble long_data
long_data <- playlist_data %>% select(-track_name, -artist) %>%
             pivot_longer(-track_id, names_to = "name", values_to = "value")

# Plot the distribution
long_data %>%
  ggplot(aes(x = value)) +
  geom_histogram(bins = 30) +
  facet_wrap(~name, scales = "free") +
  theme_minimal()
```



1C) Songs in Prof B's music library

The `in_library` variable indicates whether or not a song from the class playlist was already in my music library.

i. Find out what proportion of the songs on the class playlist were already in my library. Save the result as `prop_lib`. This object should be numeric, (i.e. `is.numeric(prop_lib)` should equal `TRUE`) AND rounded to 3 decimal places.

```r
# Calculate the proportion
prop_lib = playlist_data %>% summarise(round(mean(in_library), 3)) %>%
           as.numeric()

# Check if prop_lib is a numeric value
is.numeric(prop_lib)
```

```
## [1] TRUE
```

Part 2: Modelling with the Spotify data

Fit models to the following specifications. You must choose the **appropriate** data and type/set up of the model (from the types we've learned in this class). Be careful to **name** them as instructed.

Run `summary()` and `confint` on each model. Additionally, exponentiate the confidence intervals and the the coefficient estimates using the `exp()` function, e.g., `exp(summary(modelA)$coeff[,1])`.

2A) Fit an intercept-only model, with `in_library` as the response. Call it `modelA`.

```r
# Fit an intercept-only model
modelA = glm(in_library ~ 1, family = binomial(link = "logit"),
                             data = playlist_data)
# Summary and Confidence Interval
summary(modelA)
```

```
##
## Call:
## glm(formula = in_library ~ 1, family = binomial(link = "logit"),
##     data = playlist_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7546  -0.7546  -0.7546  -0.7546   1.6705
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.11062    0.09815  -11.32   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 623.68  on 556  degrees of freedom
## Residual deviance: 623.68  on 556  degrees of freedom
## AIC: 625.68
##
## Number of Fisher Scoring iterations: 4
```

```r
confint(modelA)
```

```
## Waiting for profiling to be done...
```

```
##      2.5 %     97.5 %
## -1.3062639 -0.9212084
```

```r
exp(confint(modelA))
```

```
## Waiting for profiling to be done...
```

```
##     2.5 %    97.5 %
## 0.2708300 0.3980378
```

```r
exp(summary(modelA)$coeff[,1])
```

```
## [1] 0.3293556
```

2B) Fit a model with `in_library` as the predictor and track_popularity as the response. Call it `modelB`.

```r
# Fit a model with `in_library`
modelB = lm(track_popularity ~ in_library, data = playlist_data)
summary(modelB)
```

```
##
## Call:
## lm(formula = track_popularity ~ in_library, data = playlist_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -76.109  -8.838   3.162  13.162  37.162
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  58.8377     0.9274  63.441   <2e-16 ***
## in_library   17.2710     1.8633   9.269   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.98 on 555 degrees of freedom
## Multiple R-squared:  0.1341, Adjusted R-squared:  0.1325
## F-statistic: 85.92 on 1 and 555 DF,  p-value: < 2.2e-16
```

```r
confint(modelB)
```

```
##               2.5 %   97.5 %
## (Intercept) 57.0160 60.65942
## in_library  13.6111 20.93088
```

```r
exp(confint(modelB))
```

```
##                     2.5 %       97.5 %
## (Intercept) 5.777403e+24 2.208268e+26
## in_library  8.151264e+05 1.230732e+09
```

```r
exp(summary(modelB)$coeff[,1])
```

```
##  (Intercept)   in_library
## 3.571842e+25 3.167337e+07
```

2C) Fit a model with `in_library` as the predictor and track_popularity ranked (using`rank()` function) as the response. Call it `modelC`.

```
# Fit a model with `in_library` RANKED VER
modelC = lm(rank(track_popularity) ~ in_library, data = playlist_data)
summary(modelC)
```

```
##
## Call:
## lm(formula = rank(track_popularity) ~ in_library, data = playlist_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -396.47 -117.66    8.34  109.53  316.34
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  239.156      7.096   33.70   <2e-16 ***
## in_library   160.818     14.257   11.28   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 145.3 on 555 degrees of freedom
## Multiple R-squared:  0.1865, Adjusted R-squared:  0.185
## F-statistic: 127.2 on 1 and 555 DF,  p-value: < 2.2e-16
```

```
confint(modelC)
```

```
##                2.5 %   97.5 %
## (Intercept) 225.2173 253.0954
## in_library  132.8142 188.8224
```

```
exp(confint(modelC))
```

```
##                     2.5 %        97.5 %
## (Intercept) 6.465702e+97 8.278058e+109
## in_library  4.791626e+57  1.010483e+82
```

```
exp(summary(modelC)$coeff[,1])
```

```
##   (Intercept)    in_library
## 7.315973e+103  6.958345e+69
```

2D) Fit an intercept-only model with `valence` as the response. Call it `modelD`.

```
# Fit an intercept-only model with `valence`
modelD = lm(valence ~ 1, data = playlist_data)
summary(modelD)
```

```
##
## Call:
## lm(formula = valence ~ 1, data = playlist_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.49103 -0.20003 -0.00603  0.19497  0.48197
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.49103    0.01048   46.85   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2474 on 556 degrees of freedom
```

```
confint(modelD)
```

```
##                 2.5 %    97.5 %
## (Intercept) 0.4704401 0.5116134
```

```
exp(confint(modelD))
```

```
##                2.5 %  97.5 %
## (Intercept) 1.600699 1.66798
```

```
exp(summary(modelD)$coeff[,1])
```

```
## [1] 1.633993
```

2E). Fit a model with `in_library` as the response and `track_popularity`, `loudness`, `acousticness`, `instrumentalness`, and `valence` as the predictors.

.

```
# Fit a model with `in_library` as the response and multiple predictors
modelE = glm(in_library ~ track_popularity + loudness + acousticness +
                          instrumentalness + valence,
                          family = binomial(link = "logit"),
                          data = playlist_data)
summary(modelE)
```

```
##
## Call:
## glm(formula = in_library ~ track_popularity + loudness + acousticness +
##     instrumentalness + valence, family = binomial(link = "logit"),
##     data = playlist_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8656  -0.7570  -0.3888  -0.0247   3.2666
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -6.712267   0.818971  -8.196 2.49e-16 ***
## track_popularity  0.064697   0.009108   7.103 1.22e-12 ***
## loudness         -0.134701   0.039695  -3.393  0.00069 ***
## acousticness     -1.509770   0.496580  -3.040  0.00236 **
## instrumentalness -2.211590   0.951227  -2.325  0.02007 *
## valence           1.326848   0.494101   2.685  0.00724 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 623.68  on 556  degrees of freedom
## Residual deviance: 496.30  on 551  degrees of freedom
## AIC: 508.3
##
## Number of Fisher Scoring iterations: 6
```

```
confint(modelE)
```

```
## Waiting for profiling to be done...
```

```
##                       2.5 %      97.5 %
## (Intercept)      -8.38893799 -5.17202582
## track_popularity  0.04763435  0.08338326
## loudness         -0.21429363 -0.05789202
## acousticness     -2.50551749 -0.55398853
## instrumentalness -4.29115074 -0.51113534
## valence           0.36529309  2.30586817
```

```
exp(confint(modelE))
```

```
## Waiting for profiling to be done...
```

```
##                        2.5 %       97.5 %
## (Intercept)      0.0002273686  0.005673065
## track_popularity 1.0487870976  1.086958311
## loudness         0.8071113580  0.943751844
## acousticness     0.0816333425  0.574653210
## instrumentalness 0.0136891635  0.599814198
## valence          1.4409362641 10.032884746
```

```
exp(summary(modelE)$coeff[,1])
```

```
##      (Intercept) track_popularity         loudness     acousticness
##      0.001215904      1.066835522      0.873977542      0.220960849
## instrumentalness          valence
##      0.109526374      3.769145967
```

2F) Songs I *should* add to my library?

Starting with the `playlist_data`, create a new tibble called `to_add`. The goal is the find the top 3 songs that *I don't currently have* in my library, that have the highest **predicted probability** of being in my library. `predict.glm` behaves similarly to `predict` function applications you've seen before. A nice feature is that adding the argument `type = "response"`, puts the predicted values on the 'response scale' (instead of the default, with is on a scale based on the link function transformation).

Arrange the dataset be descending `predicted_val` (i.e., biggest values as the top, smallest at the bottom) and then choose only the first three songs. I.e., your final tibble should only have three rows.

Here is a template code structure I recommend.

```
to_add <- ---------- %>%
  mutate(predicted_val = predict.glm(modelE, type = "response")) %>%
  ---------- %>%
  ---------- %>%
  ----------
```

```
# Create to_add tibble
to_add <- playlist_data %>%
```

```
  mutate(predicted_val = predict.glm(modelE, type = "response")) %>%
  filter(in_library == 0) %>%
  arrange(desc(predicted_val)) %>%
  head(3)
```

Part 3: Play counts

3A) Data preparation and summary

Suppose I also had data on how many times I had played each of the songs from the class playlist that were also in my own Spotify library. (This is actually a lie, as I don't use Spotify day-to-day!)

i. Load the `play_count.csv` data from the data folder and call the tibble `play_count`.
ii. Create a new tibble called `play_count_full`, and starting with the `play_count` data, merge on the `playlist_data`. Note: Choose an appropriate join function, so that no songs for which there is not play count data are included.
iii. Create two new variables: `valence_cat` and `energy_cat`, that dichotomize each of these variables into "High" and "Low" with high values being 0.5 or above, and low below 0.5.
iv. Create `count_summary`, that summarizes `play_count_full` to find the mean and variance for `play_count` and the number of observations, in each combination of the levels of `valence_cat` and `energy_cat` (i.e., there should be four rows). Add `,.groups = "drop"as an argument tosummarize`.

```
# Load data
play_count = read_csv("data/play_count.csv")
```

```
## Rows: 138 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): track_id
## dbl (1): play_count
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Create tibble `play_count_full`
play_count_full <- play_count %>% left_join(playlist_data, by = "track_id") %>%
    mutate(valence_cat = case_when(valence >= 0.5 ~ "High",
                                   valence < 0.5 ~ "Low")) %>%
    mutate(energy_cat = case_when(energy >= 0.5 ~ "High",
                                  energy < 0.5 ~ "Low"))

# Create `count_summary`
count_summary <- play_count_full %>% group_by(valence_cat, energy_cat) %>%
                 summarise(mean = mean(play_count), var = var(play_count),
                           n = n(), .groups = "drop")
```

3B) Modelling play counts

i. Use Poisson regression to fit a model with `play_count` as the response and the main effects of `valence_cat` and `energy_cat` (i.e., not interactions). Call it `model_counts` and get the summary and confidence intervals for it. Additionally, get the exponentiated version of the summary and confint results (you can use the `exp()` function to do this. You'll want to extract the coefficient table from the summary object with something like this: `exp(summary(model_counts)$coeff[,1])`.

```
# Fit model required
model_counts = glm(play_count ~ valence_cat + energy_cat,
                   family = poisson, data = play_count_full)
summary(model_counts)
```

```
##
## Call:
## glm(formula = play_count ~ valence_cat + energy_cat, family = poisson,
```

```
##     data = play_count_full)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -4.825  -3.024  -0.691   1.212  14.258
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     2.45455    0.03337  73.545  < 2e-16 ***
## valence_catLow -1.30952    0.09227 -14.192  < 2e-16 ***
## energy_catLow  -0.83906    0.12764  -6.573 4.92e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1822.9  on 137  degrees of freedom
## Residual deviance: 1424.7  on 135  degrees of freedom
## AIC: 1817.1
##
## Number of Fisher Scoring iterations: 6
```

```
confint(model_counts)
```

```
## Waiting for profiling to be done...
```

```
##                    2.5 %     97.5 %
## (Intercept)     2.388424  2.5192653
## valence_catLow -1.494422 -1.1324421
## energy_catLow  -1.098390 -0.5972458
```

```
exp(confint(model_counts))
```

```
## Waiting for profiling to be done...
```

```
##                     2.5 %     97.5 %
## (Intercept)    10.8963048 12.4194685
## valence_catLow  0.2243782  0.3222453
## energy_catLow   0.3334075  0.5503253
```

```
exp(summary(model_counts)$coeff[,1])
```

```
##    (Intercept) valence_catLow  energy_catLow
##     11.6411441      0.2699490      0.4321172
```

Part 4: Music and stress

It is often suggested that music can help with stress relief.

According to Labbé et al (2007), "music is an important aspect of youth culture" and that young people will "often have a collection of favourite 'tunes' that they will listen to when they are feeling 'stressed out' ". (I include these quotes because they amuse me. . . they kinda seem like a parody of how academics might write about the 'youth').

Anyways, suppose you wished to do some research about how listening to music impacts study stress. Your idea is that listening to calming music while taking a timed quiz will result in higher grades than listening to silence will.

4A) Pilot study

Suppose you wanted to do a small informal pilot of this study to get a sense of how to run the protocol.

You and 3 of your friends worked together to write and share practice quizzes to help you prepare for the STA303/1002 mixed assessment.

i. Read in the `pilot_data.csv` data (in the `data` folder) and save it as `pilot_data`.

ii. Fit 3 appropriate models, `mod_person`, `mod_condition` and `mod_both`, where each model has score as the response and then person, condition and both person and condition as the predictors, respectively (no interactions).

iii. Get the model matrix for `mod_both` and save it as `mod_both_mat`.

iv. Run the following three likelihood ratio tests once you have created the models.

```
test1 <- lrtest(mod_person, mod_both)
test2 <- lrtest(mod_condition, mod_both)
test3 <- lrtest(mod_condition, mod_person)
```

```
# Load data
pilot_data <- read_csv("data/pilot_data.csv")
```

```
## Rows: 12 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): person, quiz, condition
## dbl (1): score
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Fit model: mod_person
mod_person = lm(score ~ person, data = pilot_data)

# Fit model: mod_condition
mod_condition = lm(score ~ condition, data = pilot_data)

# Fit model: mod_both
mod_both = lm(score ~ person + condition, data = pilot_data)

# Save model matrix for `mod_both`
mod_both_mat = model.matrix(mod_both)

# Run tests
test1 <- lrtest(mod_person, mod_both)
```

```
test2 <- lrtest(mod_condition, mod_both)
test3 <- lrtest(mod_condition, mod_person)
```

4B) Propsing a larger study (reading only)

You want to pitch a larger study to be run as part of STA303/1002 in 2023. Your idea is that students will be randomized and required to either listen to calming classical music or silence while doing the 'pre-knowledge quiz'.

You know that students will need to provide consent if you want to use their data for publication. Your suggestion to ensure that students will agree to take part is that Prof. Bolton makes it her polocy that she will only write reference letters for students who consent to take part in the study.

References

Labbé, E., Schmidt, N., Babin, J., & Pharr. M. *Coping with Stress: The Effectiveness of Different Types of Music*. Appl Psychophysiol Biofeedback 32, 163–168 (2007). https://doi.org/10.1007/s10484-007-9043-9

Part 5: Web scraping

Suppose you also wanted to get access to the data about the songs on the class playlist BUT you don't have a Spotify account [1], which is required to use the `spotifyR` package.

You are, of course, committed to only being an ethical scraper.

i) Start by checking the relevant contents of the robots.txt with the `polite` package. Ensure you have updated the code below appropriately to be aligned with ethical practice. The object `session` will be autograded.

ii) Study prompt: Should you check any additional information from Spotify to ensure you are being an ethical scraper? If yes, please do so.

```
# I have set cache = TRUE for this chunk as you don't want to run it
# EVERY time you knit

# Update the following code appropriately for your needs

# URL for our class playlist
url <- "https://open.spotify.com/playlist/7wnSMzB7ZrFBHODG0ycJIV"
session <- bow(url, user_agent =
  "Mixed assessment for STA303/1002H1, jiahao.bai@mail.utoronto.ca")
session

# Tip: The Team Up! activity from Module 2 has an example of using the bow function
```

[1] If you do, pretend you don't.

Checklist

- Updated name and student ID number on lines 4 and 12 (i.e., in the YAML)
- Other than the `setup` chunk (where the packages are installed), **all chunks should have eval=TRUE in the chunk options** (you don't usually need to say this explicitly in each chunk option, but it makes the autograding simpler)
- To make things easier during the timed component, **make sure you have summaries and confidence intervals produced for all models** and show objects in your PDF (none of the outputs should be too long, i.e., don't print raw datasets, etc.)
- **Models**
  - `modelA`
  - `modelB`
  - `modelC`
  - `modelD`
  - `modelE`
  - `model_counts`
  - `mod_person`
  - `mod_condition`
  - `mod_both`
- **Tibbles**
  - `playlist_data`
  - `long_data`, with columns `track_id`, `name`, `value`
  - `to_add`
  - `play_count`
  - `play_count_full`
  - `count_summary`
  - `pilot_data`
- **Values/other objects**
  - `prop_lib`
  - `mod_both_mat` (note, this is a matrix, not a tibble)
  - `test1`
  - `test2`
  - `test3`
  - `url` (code already provided)
  - `session`, the result of the bow function
- PDF knit from RMD (directly, no interim HTML/Word step)
- Rmd and PDF files names correctly.
- Rmd and PDF files submitted to MarkUs BEFORE 8:00 p.m. ET