Modern Pop Album Analysis using Spotify API

Jenny Eberling

2022-05-03

Uploading Data

```
# useful packages
library(dplyr)
library(spotifyr)
library(plotly)
library(ggplot2)
library(stringr) #to take out acoustic versions
library(knitr) # to be able to kable
library(tidyr) # for 'taken from class' section
library(GGally) #for ggpairs
library(ipred) # for bagging
library(factoextra) # for 'fun cluster graph in spotify_learning
library(MASS) # QDA
library(e1071) # Naive Bayes
library(rpart) # classification tree
library(partykit) # pretty classification tree
library(caret) # VarImp
library(randomForest)
library(corrplot) # for fun correlation plot
library(gridExtra) # for log plots


songs <- readRDS("C:/Users/Owner/OneDrive/Desktop/Semester IV/Data
Mining/Project/pop_songs.rds")

### Serious data mining

songs_num <- songs %>% ungroup() %>%

dplyr::select(c("danceability","energy","key","loudness","mode","speechiness"
,"acousticness","instrumentalness","liveness","valence","tempo","explicit","d
uration_min","aoty","album_length"))

# ignore track_number and third to compare which songs are diff
songs_pred <- songs %>% ungroup() %>%

dplyr::select(c("danceability","energy","key","loudness","mode","speechiness"
,"acousticness","instrumentalness","liveness","valence","tempo","explicit","d
uration_min","aoty","album_length","song_third_num"))
```

```r
# Same as above but with appropriate logged variables
# can't include log(instrumentalness) because too many -inf values when inst
= 0
songs_numlog <- songs %>% ungroup() %>%

dplyr::select(c("danceability","energy","key","loudness","mode","speechiness"
,"acousticness","instrumentalness","liveness","valence","tempo","explicit","d
uration_min","aoty","album_length")) %>%
  mutate(speechinesslog = log(speechiness),
         livenesslog = log(liveness))
songs_numlog <- songs_numlog %>%

dplyr::select(c("danceability","energy","key","loudness","mode","speechinessl
og","acousticness","instrumentalness","livenesslog","valence","tempo","explic
it","duration_min","aoty","album_length"))

songs_predlog <- songs %>% ungroup() %>%

dplyr::select(c("danceability","energy","key","loudness","mode","speechiness"
,"acousticness","instrumentalness","liveness","valence","tempo","explicit","d
uration_min","aoty","album_length","song_third_num")) %>%
  mutate(speechinesslog = log(speechiness),
         livenesslog = log(liveness))

songs_predlog <- songs_predlog %>%

dplyr::select(c("danceability","energy","key","loudness","mode","speechinessl
og","acousticness","instrumentalness","livenesslog","valence","tempo","explic
it","duration_min","aoty","album_length","song_third_num"))


songs2 <- songs %>%
  mutate(third = ifelse(songs$song_third == "third",1,0))

# Songs but with logged vars
songs_log <- songs %>%
  mutate(livenesslog = log(liveness),
         speechinesslog = log(speechiness)) %>%
  dplyr::select(-c(liveness,speechiness))



########################################
# Predicting if songs are on aoty or not
########################################
grammy <- songs %>% group_by(album_name) %>%
  summarize(danceability = mean(danceability),
            energy = mean(energy),
            loudness = mean(loudness),
```

```
            mode = mean(mode),
            speechiness = mean(speechiness),
            acousticness = mean(acousticness),
            instrumentalness = mean(instrumentalness),
            liveness = mean(liveness),
            valence = mean(valence),
            tempo = mean(tempo),
            duration_min = mean(duration_min),
            explicit = ifelse(mean(explicit) > 0, 1,0),
            aoty = mean(aoty),
            album_length = mean(album_length))

# with logged variables
grammylog <- songs %>% group_by(album_name) %>%
  summarize(danceability = mean(danceability),
            energy = mean(energy),
            loudness = mean(loudness),
            mode = mean(mode),
            speechinesslog = mean(log(speechiness)),
            acousticness = mean(acousticness),
            instrumentalnesslog = mean(log(instrumentalness)),
            livenesslog = mean(log(liveness)),
            valence = mean(valence),
            tempo = mean(tempo),
            duration_min = mean(duration_min),
            explicit = ifelse(mean(explicit) > 0, 1,0),
            aoty = mean(aoty),
            album_length = mean(album_length))
```

Creating the data - do not run

```
# getting setup
Sys.setenv(SPOTIFY_CLIENT_ID = 'f77ab842e2074e4085981927b265c1b4')
Sys.setenv(SPOTIFY_CLIENT_SECRET = 'd6d020e996e145c4aa5df8beac578d95')
access_token <- get_spotify_access_token()
my_id <- 'qrovmuqylvycmj3vsnqk57qlj'
#my_plists <- get_user_playlists(my_id)

############ Swift #############
# the iconic track 5 - is it really different?
# " The fifth track is always the most emotional, tear-jerking and vulnerable
song on the album. "
swift0 <- get_artist_audio_features('taylor swift', include_groups =
c("album","single")) # not including appears_on
singles <- swift0 %>% filter(track_name == "Christmas Tree Farm") # later
I'll probably throw this into the big filter and consolidate later.

unique(swift0$album_name)

# cleaning Swift
```

```
swift1 <- swift0 %>%
  filter(swift0$album_name %in% c("Taylor Swift", "Speak Now", "Fearless",
                                  "Red","1989",
                                  "reputation", "Lover", "folklore",
                                  "evermore")) %>%

select(c("artist_name","album_release_year","danceability","energy","key","lo
udness",

"mode","speechiness","acousticness","instrumentalness","liveness","valence",
           "tempo","time_signature","duration_ms",
"explicit","track_name","track_number","album_name",
           "key_name","mode_name","key_mode"))

swift <- swift1 %>%
  filter((track_name == "Our Song" & album_name == "Fearless") == FALSE) %>%
  filter((track_name == "Should've Said No" & album_name == "Fearless") ==
FALSE) %>%
  group_by(track_name) %>% filter(row_number(track_name) == 1) %>%
  filter(track_name != "Love Story - J Stax Radio Mix") %>%
  filter(track_name != "Picture To Burn - Radio Edit") %>%
  filter(str_detect(track_name, "Karaoke") == FALSE) %>%
  filter(track_name != "Invisible") %>%
  filter(track_name != "Teardrops On My Guitar") %>%
  filter(track_name != "A Perfectly Good Heart") %>%
  filter(track_name != "Teardrops on My Guitar - Pop Version") %>%
  filter(track_name != "Beautiful Eyes") %>%
  filter(track_name != "I'm Only Me When I'm With You") %>%
  filter(track_name != "I Heart ?") %>%
  filter(track_name != "Mine - POP Mix") %>%
  mutate(duration_min = duration_ms/1000/60)

swift$album = as.factor(swift$album_name)
levels(swift$album)
# Nickname the albums
levels(swift$album) <-
c('1989','evermore','Fearless','folklore','Lover','Red','reputation',
                        'Speak Now', 'Debut')
# Now in order
swift$album <- factor(swift$album, levels = c("Debut", "Fearless", "Speak
Now",
                                              "Red", "1989", "reputation",
"Lover",
                                              "folklore","evermore"))
table(swift$album)

############ Adele #############
adele0 <- get_artist_audio_features('adele', include_groups =
c("album","single")) # not including appears_on
singles[2,] <- adele0 %>% filter(track_name == "Skyfall") # an adele single
```

```
adele1 <- adele0 %>%
  filter(adele0$album_name %in% c("19", "21", "25","30")) %>%

select(c("artist_name","album_release_year","danceability","energy","key","lo
udness",

"mode","speechiness","acousticness","instrumentalness","liveness","valence",

"tempo","time_signature","duration_ms","explicit","track_name","track_number"
,"album_name",
           "key_name","mode_name","key_mode"))

adele <- adele1 %>%
  group_by(track_name) %>% filter(row_number(track_name) == 1) %>%
  filter(track_name != "Turning Tables - Live Acoustic")%>%
  filter(track_name != "Don't You Remember - Live Acoustic")%>%
  filter(track_name != "Someone Like You - Live Acoustic") %>%
  filter(!str_detect(track_name, "Live at Hotel Cafe")) %>%
  filter(!str_detect(track_name, "Many Shades Of Black - Performed by The
Raconteurs")) %>%
  mutate(duration_min = duration_ms/1000/60)

table(adele$album_name)

adele$album = as.factor(adele$album_name)
levels(adele$album)


############ Dua Lipa #############
lipa0 <- get_artist_audio_features('dua lipa', include_groups =
c("album","single")) # not including appears_on
singles[3,] <- lipa0 %>% filter(track_name == "UN DIA (ONE DAY) (Feat.
Tainy)")


lipa1 <- lipa0 %>%
  filter(lipa0$album_name %in% c("Dua Lipa", "Future Nostalgia")) %>%

select(c("artist_name","album_release_year","danceability","energy","key","lo
udness",

"mode","speechiness","acousticness","instrumentalness","liveness","valence",

"tempo","time_signature","duration_ms","explicit","track_name","track_number"
,
           "album_name","key_name","mode_name","key_mode"))

lipa <- lipa1 %>%
```

```
  group_by(track_name) %>% filter(row_number(track_name) == 1) %>%
  filter(track_name != "Fever (feat. Angèle)") %>%
  mutate(duration_min = duration_ms/1000/60)


lipa$album = as.factor(lipa$album_name)
levels(lipa$album)
# Nickname the albums
levels(lipa$album) <- c('Debut', 'Future Nos')

table(lipa$album)


############ Carly Rae Jepsen #############
jepsen0 <- get_artist_audio_features('carly rae jepsen', include_groups =
c("album","single")) # not including appears_on
singles[4,] <- jepsen0 %>% filter(track_id == "6tcjcJFRSdkIlThUUP39mJ") #
"let's be friends"

jepsen1 <- jepsen0 %>%
  filter(jepsen0$album_name %in% c("Tug Of War", "Kiss", "Emotion","EMOTION
SIDE B",
                                   "Dedicated","Dedicated Side B")) %>%

select(c("artist_name","album_release_year","danceability","energy","key","lo
udness",

"mode","speechiness","acousticness","instrumentalness","liveness","valence",

"tempo","time_signature","duration_ms","explicit","track_name","track_number"
,"album_name",
           "key_name","mode_name","key_mode"))

jepsen <- jepsen1 %>%
  group_by(track_name) %>% filter(row_number(track_name) == 1) %>%
  filter(track_name != "Almost Said It") %>%
  filter(track_name != "Melt With You") %>%
  filter(track_name != "Picture") %>%
  filter((album_name == "Emotion" & track_number > 12) == FALSE) %>%
  mutate(duration_min = duration_ms/1000/60)

table(jepsen$album_name)

jepsen$album = as.factor(jepsen$album_name)
levels(jepsen$album)
# Nickname the albums
levels(jepsen$album) <- c("Dedicated", "Dedic B","Emotion","Emotion
B","Kiss","ToW")
# Now in order
```

```r
jepsen$album <- factor(jepsen$album, levels =
c("ToW","Kiss","Emotion","Emotion B",
                                                "Dedicated","Dedic B"))


table(jepsen$album)


############ Justin Bieber #############
bieber0 <- get_artist_audio_features('justin bieber', include_groups =
c("album","single")) # not including appears_on
singles[5,] <- bieber0 %>% filter(track_name == "Attention")


bieber1 <- bieber0 %>%
  filter(bieber0$album_name %in% c("My World","My World 2.0", "Under The
Mistletoe", "Believe",
                                   "Journals", "Purpose (Deluxe)",
"Changes","Justice")) %>%

select(c("artist_name","album_release_year","danceability","energy","key","lo
udness",

"mode","speechiness","acousticness","instrumentalness","liveness","valence",

"tempo","time_signature","duration_ms","explicit","track_name","track_number"
,"album_name",
           "key_name","mode_name","key_mode"))

bieber <- bieber1 %>%
  group_by(track_name) %>% filter(row_number(track_name) == 1) %>%
  filter(track_name != "Been You")%>% # what is a more efficient way???
  filter(track_name != "Get Used to It")%>%
  filter(track_name != "We Are") %>%
  filter(track_name != "Trust") %>%
  filter(track_name != "All In It") %>%
  filter(track_name != "Get Used To It") %>%
  filter(track_name != "Unstable (feat.The Kid LAROI)") %>%
  filter(track_name != "What Do You Mean? - Acoustic") %>%
  filter(!str_detect(track_name, "Single Version")) %>%
  mutate(duration_min = duration_ms/1000/60)


bieber$album = as.factor(bieber$album_name)
levels(bieber$album)
# Nickname the albums
levels(bieber$album) <-
c("Believe","Changes","Journals","Justice","World","World 2.0",
                          "Purpose","Mistletoe")
```

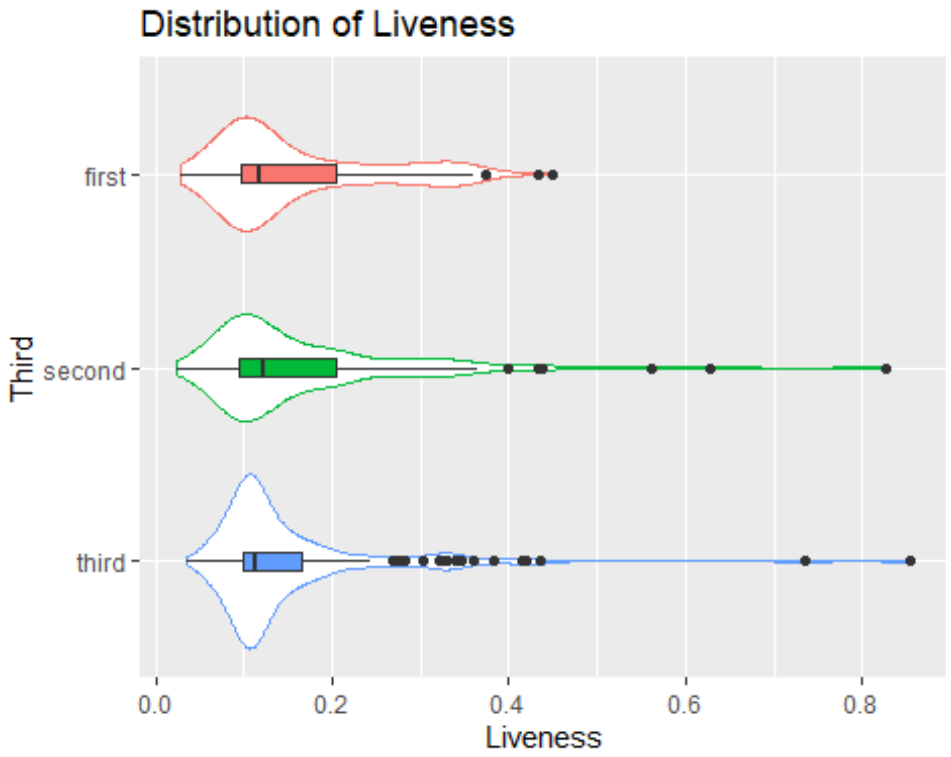
```r
# Now in order
bieber$album <- factor(bieber$album, levels = c("World","World
2.0","Mistletoe","Believe",

"Journals","Purpose","Changes","Justice"))

table(bieber$album)

############ Ariana Grande #############
grande0 <- get_artist_audio_features('ariana grande', include_groups =
c("album","single")) # not including appears_on
singles[6,] <- grande0 %>% filter(track_id == "4HBZA5flZLE435QTztThqH") #
"stuck with u"

grande1 <- grande0 %>%
  filter(grande0$album_name %in% c("Yours Truly", "My Everything","Dangerous
Woman",
                                   "Sweetener","thank u, next","Positions"))
%>%

select(c("artist_name","album_release_year","danceability","energy","key","lo
udness",

"mode","speechiness","acousticness","instrumentalness","liveness","valence",

"tempo","time_signature","duration_ms","explicit","track_name","track_number"
,"album_name",
           "key_name","mode_name","key_mode"))


grande2 <- grande1 %>%
  group_by(track_name) %>%
  filter(row_number(track_name) == 1) %>%
  filter(track_name != "Step On Up")%>%
  filter(track_name != "Focus")%>%
  filter(track_name != "Jason's Song (Gave It Away)")%>%
  filter(track_name != "Bang Bang") %>%
  filter(track_name != "Only 1")%>%
  filter(track_name != "You Don't Know Me")%>%
  filter(track_name != "Cadillac Song")%>%
  filter(track_name != "Too Close")%>%
  filter(track_name != "Baby I") %>%
  filter(track_name != "Baby I - Cosmic Dawn Radio Edit") %>%
  filter(track_name != "Right There - 7th Heaven Radio Edit") %>%
  filter((track_name == "thank u, next" & track_number =="1") == FALSE) %>% #
a single got on there
  mutate(duration_min = duration_ms/1000/60)

# There is one song that is getting put on the wrong album
```

```r
baby_I <- grande0 %>% dplyr::filter(track_id == "4TQYyBORpYzICpQtzINUbg") %>%
  dplyr::filter(album_name %in% c("Yours Truly", "My Everything","Dangerous
Woman",
                                  "Sweetener","thank u, next","Positions"))
%>%

dplyr::select(c("artist_name","album_release_year","danceability","energy","k
ey","loudness",

"mode","speechiness","acousticness","instrumentalness","liveness","valence",

"tempo","time_signature","duration_ms","explicit","track_name","track_number"
,"album_name",
           "key_name","mode_name","key_mode")) %>%
  mutate(duration_min = duration_ms/1000/60)

grande <- rbind(grande2,baby_I)


grande$album = as.factor(grande$album_name)
levels(grande$album)

# Now in order
grande$album <- factor(grande$album, levels = c("Yours Truly","My
Everything",
                                                "Dangerous
Woman","Sweetener",
                                                "thank u, next","Positions"))
grande <- grande %>%
  group_by(album_name) %>%
  arrange(track_number, .by_group = TRUE)

############ Merge them #############
#swift$artist_name = "Taylor Swift"
#adele$artist_name = "Adele"

songs0 <- rbind(swift, adele, lipa, jepsen, bieber, grande)

# identify which ones won album of the year
songs0$aoty <- ifelse(songs0$album_name %in%
c("folklore","25","1989","21","Fearless","Purpose (Deluxe)"),1,0)


# which third of the album is it in
songs1 <- songs0 %>%
  group_by(album_name) %>%
  mutate(album_length = n()) %>%
  mutate(track_placement = (track_number-.5)/album_length) %>%
  ungroup() %>%
```

```
  mutate(song_third = ifelse(track_number/album_length > 2/3, "third",
                             ifelse(track_number/album_length > 1/3,"second",
"first")))

songs <- songs1 %>% ungroup() %>%
  mutate(song_third_num = ifelse(song_third == 'first',1,
                                 ifelse(song_third == 'second',2,3)))


table(songs$song_third,songs$album_length)

table(songs$artist_name)
table(swift$album)
table(bieber$album)
table(lipa$album)
table(jepsen$album)
table(grande$album)

saveRDS(adele, file = "adele.rds")
saveRDS(bieber, file = "bieber.rds")
saveRDS(grande, file = "grande.rds")
saveRDS(jepsen, file = "jepsen.rds")
saveRDS(lipa, file = "lipa.rds")
saveRDS(swift, file = "swift.rds")

saveRDS(songs, file = "pop_songs.rds")
saveRDS(singles, file = "pop_singles.rds")
```

Data Exploration

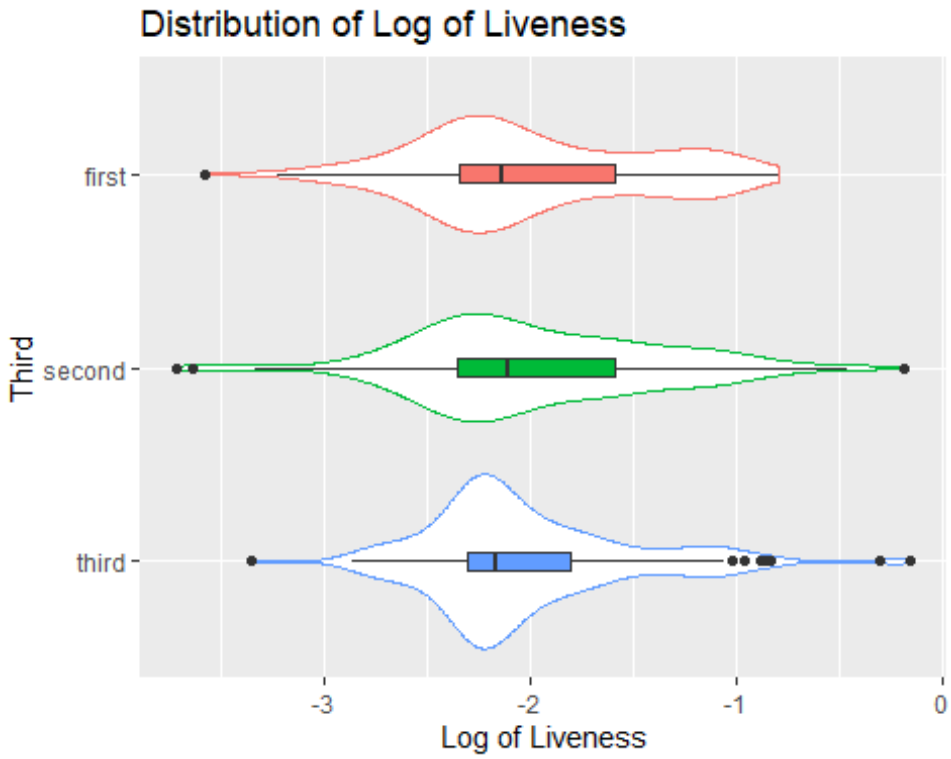
```
#################################################################
# Data Exploration by third
#################################################################

# Liveness - second third is most 'live', high outliers in second and third
ggplot(songs, aes(x = factor(song_third, level = c('third', 'second',
'first')), liveness)) +
  geom_violin(aes(color = song_third)) +
  geom_boxplot(aes(fill = song_third), width = 0.1) +
  guides(color = F, fill = F) +     # removes legends for color and fill
aesthetics
  labs (title = "Distribution of Liveness",
        y = "Liveness",
        x = "Third") +
  coord_flip()
```



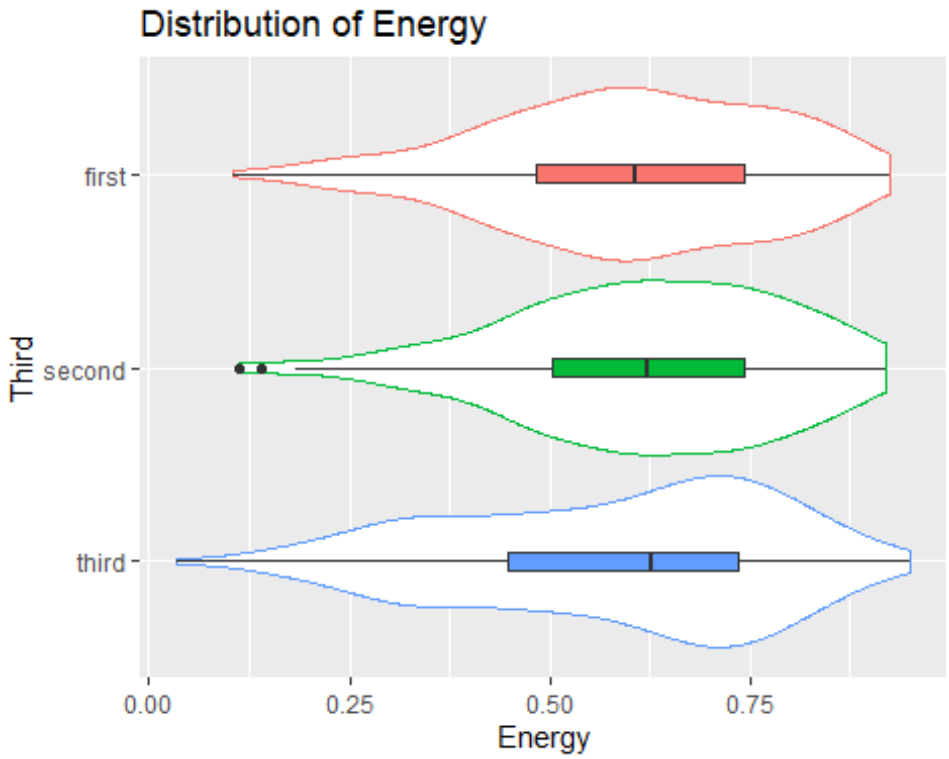

```
songs %>% group_by(song_third) %>% summarize(mean(liveness))
```

```
## # A tibble: 3 x 2
##   song_third `mean(liveness)`
##   <chr>                 <dbl>
## 1 first                 0.158
## 2 second                0.166
## 3 third                 0.152
```

```
# Log of Liveness
ggplot(songs, aes(x = factor(song_third, level = c('third', 'second',
'first')), log(liveness))) +
  geom_violin(aes(color = song_third)) +
  geom_boxplot(aes(fill = song_third), width = 0.1) +
  guides(color = F, fill = F) +     # removes legends for color and fill
aesthetics
  labs (title = "Distribution of Log of Liveness",
        y = "Log of Liveness",
        x = "Third") +
  coord_flip()
```



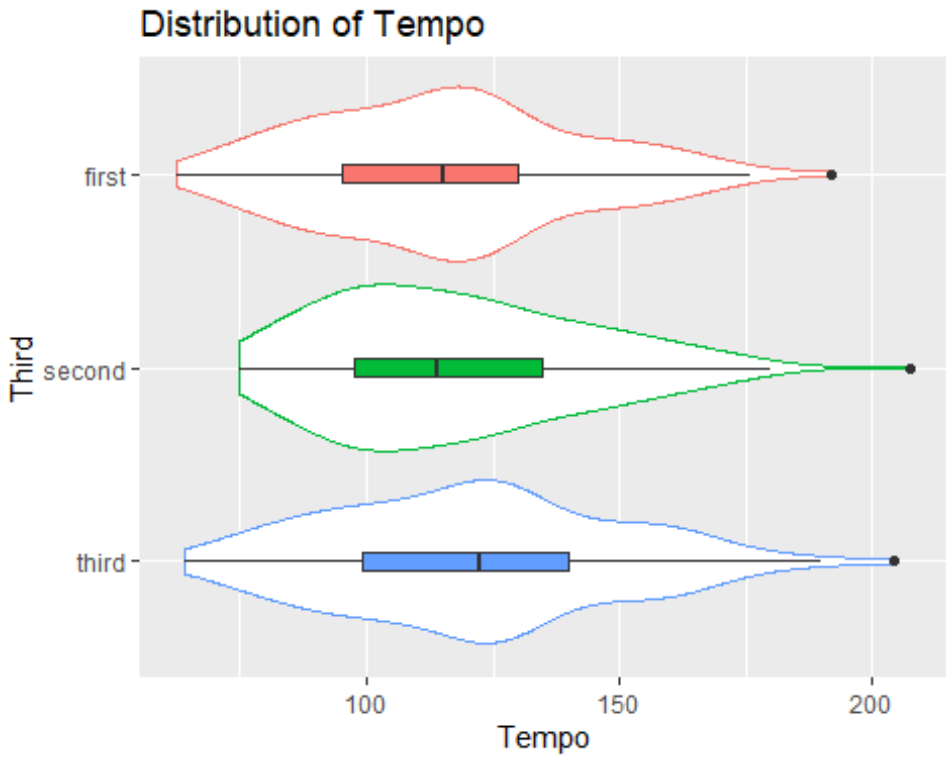

```
# Energy - second is highest energy, then first
ggplot(songs, aes(x = factor(song_third, level = c('third', 'second',
'first')), energy)) +
  geom_violin(aes(color = song_third)) +
  geom_boxplot(aes(fill = song_third), width = 0.1) +
  guides(color = F, fill = F) +     # removes legends for color and fill
aesthetics
  labs (title = "Distribution of Energy",
        y = "Energy",
        x = "Third") +
  coord_flip()
```




```
songs %>% group_by(song_third) %>% summarize(mean(energy))
```

```
## # A tibble: 3 x 2
##   song_third `mean(energy)`
##   <chr>                <dbl>
## 1 first                0.602
## 2 second               0.620
## 3 third                0.579
```

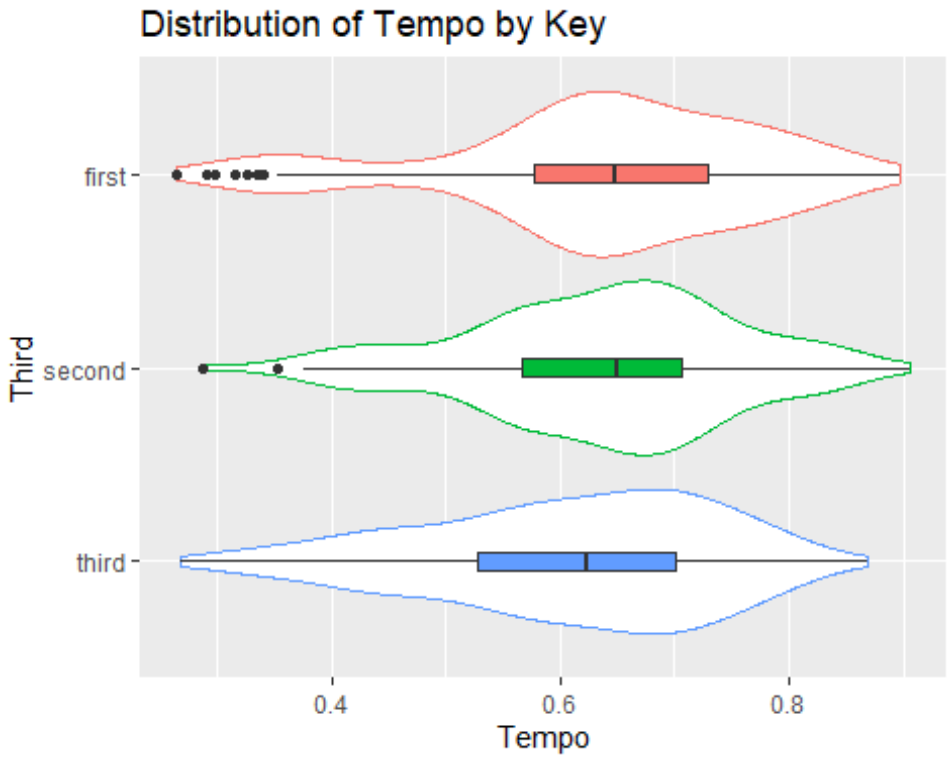
```
# Tempo - picks up as album goes through
ggplot(songs, aes(x = factor(song_third, level = c('third', 'second',
'first')), tempo)) +
  geom_violin(aes(color = song_third)) +
  geom_boxplot(aes(fill = song_third), width = 0.1) +
  guides(color = F, fill = F) +     # removes legends for color and fill
aesthetics
  labs (title = "Distribution of Tempo",
        y = "Tempo",
        x = "Third") +
  coord_flip()
```



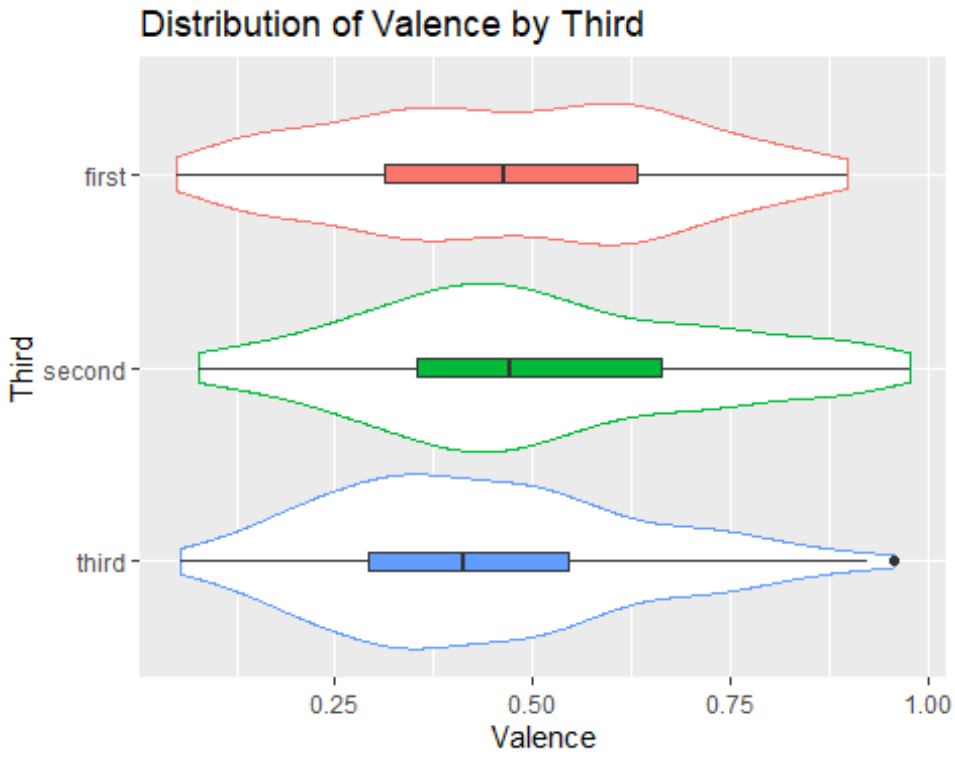

```
songs %>% group_by(song_third) %>% summarize(mean(tempo))
```

```
## # A tibble: 3 x 2
##   song_third `mean(tempo)`
##   <chr>              <dbl>
## 1 first               115.
## 2 second              118.
## 3 third               121.
```

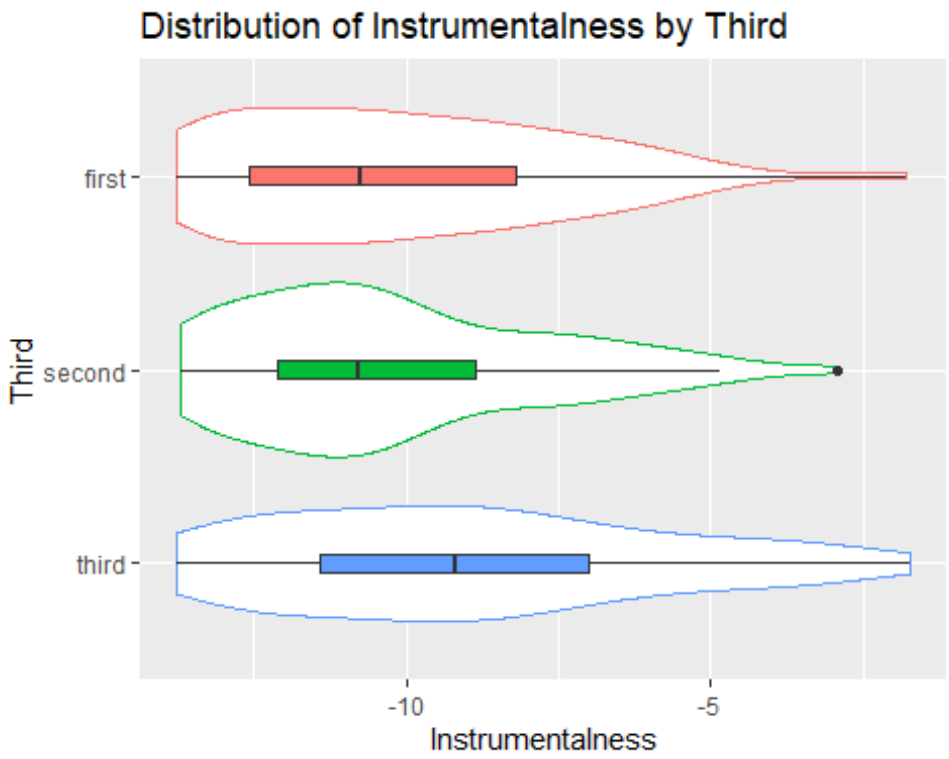
```
# Tempo vs Third - thirds get less danceable
ggplot(songs, aes(x = factor(song_third, level = c('third', 'second',
'first')), danceability)) +
  geom_violin(aes(color = song_third)) +
  geom_boxplot(aes(fill = song_third), width = 0.1) +
  guides(color = F, fill = F) +     # removes legends for color and fill
aesthetics
  labs (title = "Distribution of Tempo by Key",
        y = "Tempo",
        x = "Third") +
  coord_flip()
```



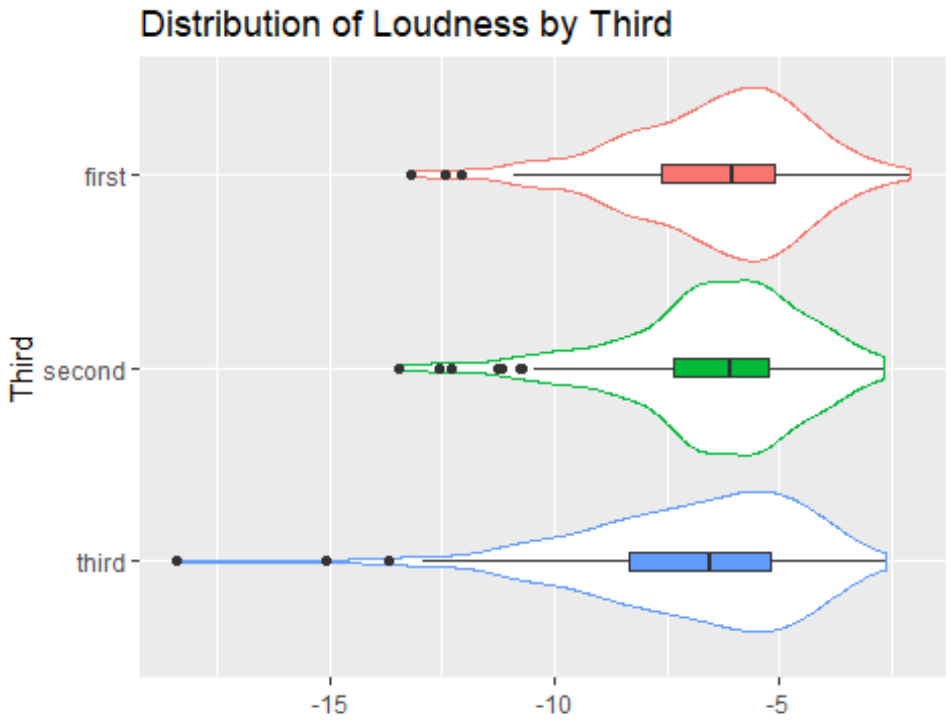

```
# Since 1 represents a major mode and 0 is minor, third third is less "moody"
table(songs$mode,songs$song_third) %>% prop.table(2)

##
##         first    second     third
##   0 0.3541667 0.3443709 0.2727273
##   1 0.6458333 0.6556291 0.7272727

# Valence vs Third - Second third of album is happiest
songs %>%
  ggplot(aes(x = factor(song_third, level = c('third', 'second', 'first')),
valence)) +
  geom_violin(aes(color = song_third)) +
  geom_boxplot(aes(fill = song_third), width = 0.1) +
  guides(color = F, fill = F) +     # removes legends for color and fill
aesthetics
  labs (title = "Distribution of Valence by Third",
        y = "Valence",
        x = "Third") +
  coord_flip()
```




```
# Instumentalness vs Third - third is more instrumental
songs %>%
ggplot(aes(x = factor(song_third, level = c('third', 'second', 'first')),
log(instrumentalness))) +
  geom_violin(aes(color = song_third)) +
  geom_boxplot(aes(fill = song_third), width = 0.1) +
  guides(color = F, fill = F) +     # removes legends for color and fill
aesthetics
  labs (title = "Distribution of Instrumentalness by Third",
        y = "Instrumentalness",
        x = "Third") +
  coord_flip()
```

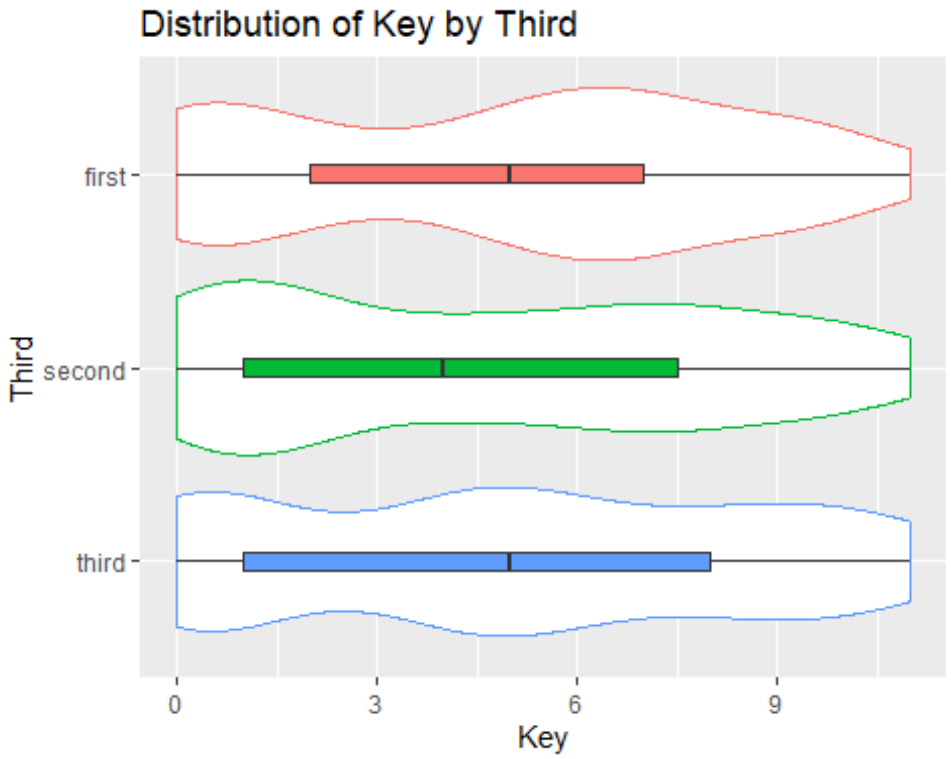



```
# Loudness vs Third - final third is less loud
ggplot(songs, aes(x = factor(song_third, level = c('third', 'second',
'first')), loudness)) +
  geom_violin(aes(color = song_third)) +
  geom_boxplot(aes(fill = song_third), width = 0.1) +
  guides(color = F, fill = F) +     # removes legends for color and fill
aesthetics
  labs (title = "Distribution of Loudness by Third",
        y = "Loudness",
        x = "Third") +
  coord_flip()
```

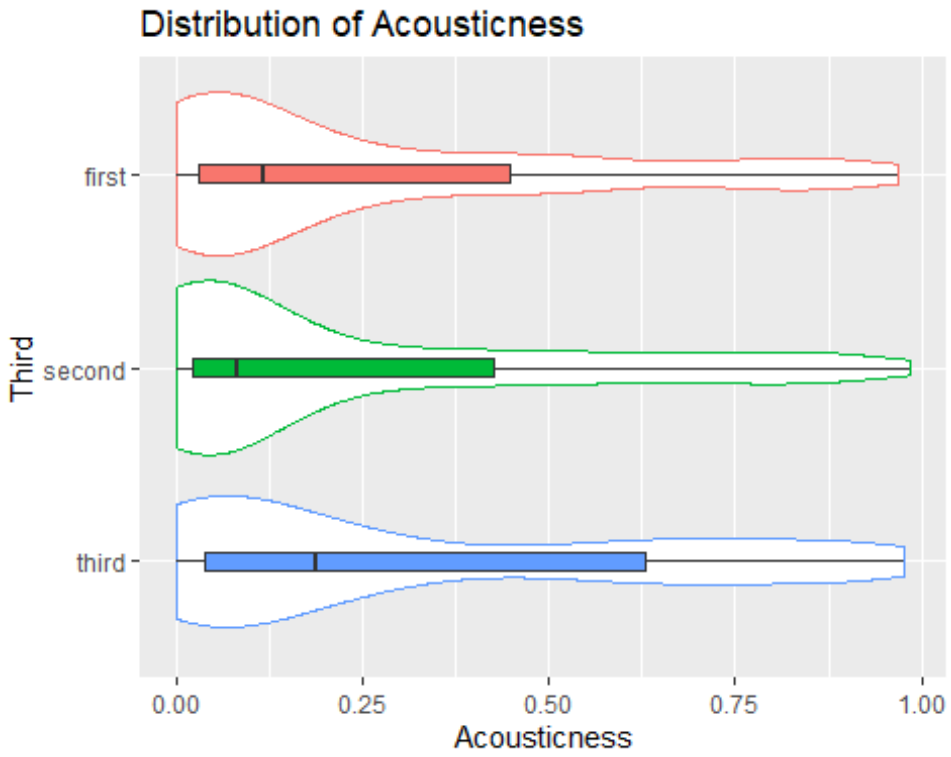


Loudness

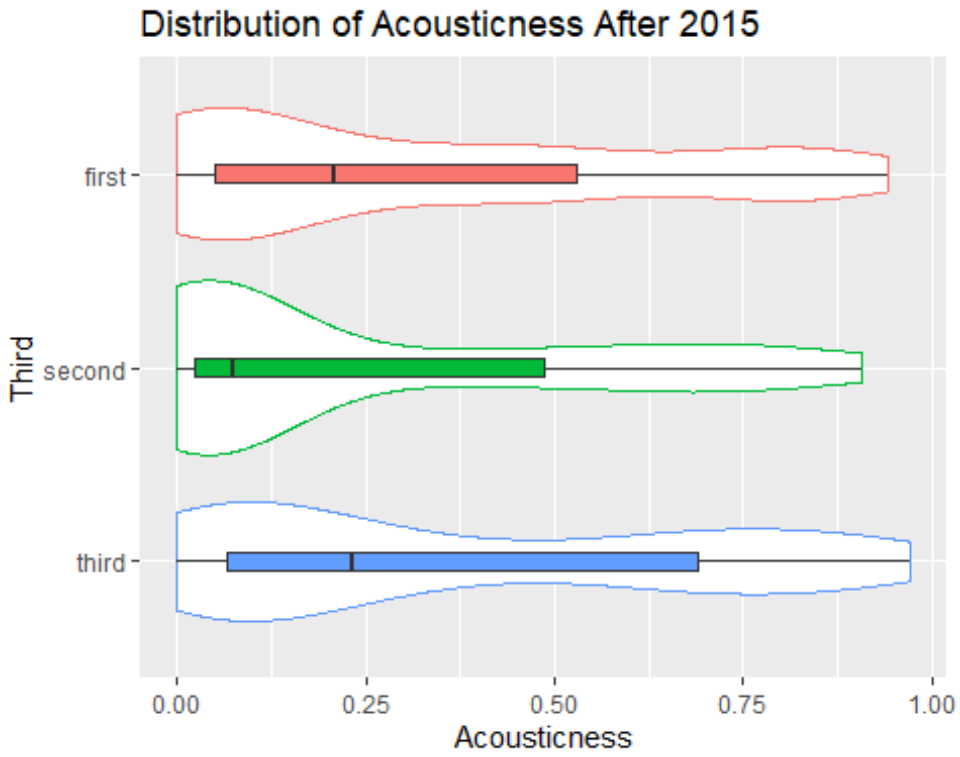
```
# Key vs Third - lower key in second part
ggplot(songs, aes(x = factor(song_third, level = c('third', 'second',
'first')), key)) +
  geom_violin(aes(color = song_third)) +
  geom_boxplot(aes(fill = song_third), width = 0.1) +
  guides(color = F, fill = F) +     # removes legends for color and fill
aesthetics
  labs (title = "Distribution of Key by Third",
        y = "Key",
        x = "Third") +
  coord_flip()
```



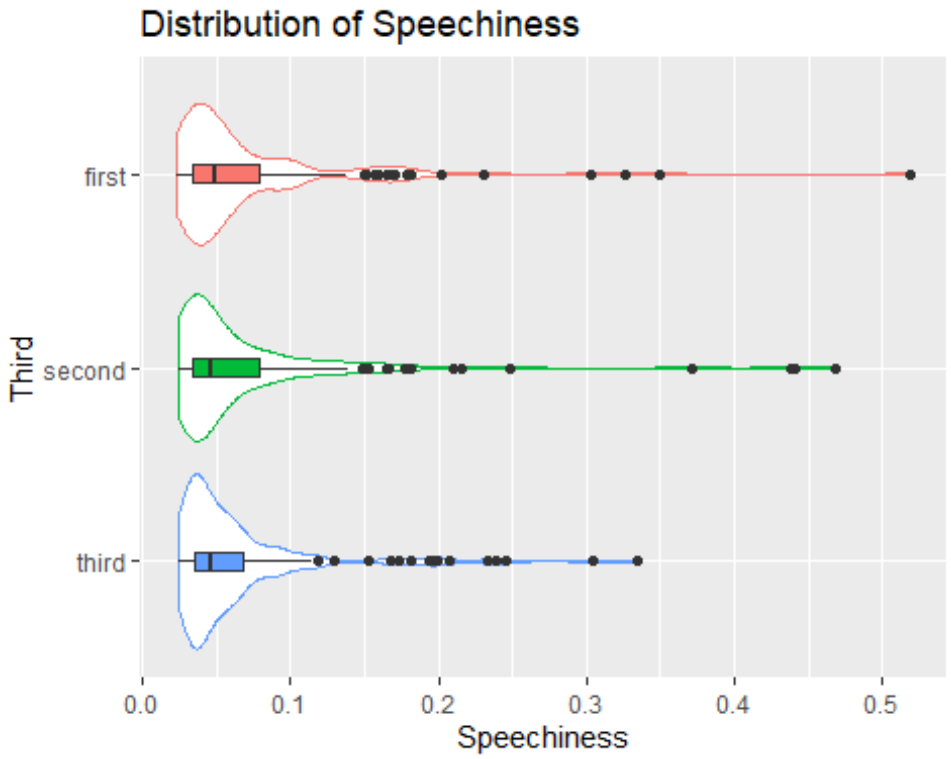

```
# Acousticness by Third - final third is more acoustic, second is the least
ggplot(songs, aes(x = factor(song_third, level = c('third', 'second',
'first')), acousticness)) +
  geom_violin(aes(color = song_third)) +
  geom_boxplot(aes(fill = song_third), width = 0.1) +
  guides(color = F, fill = F) +     # removes legends for color and fill
aesthetics
  labs (title = "Distribution of Acousticness",
        y = "Acousticness",
        x = "Third") +
  coord_flip()
```



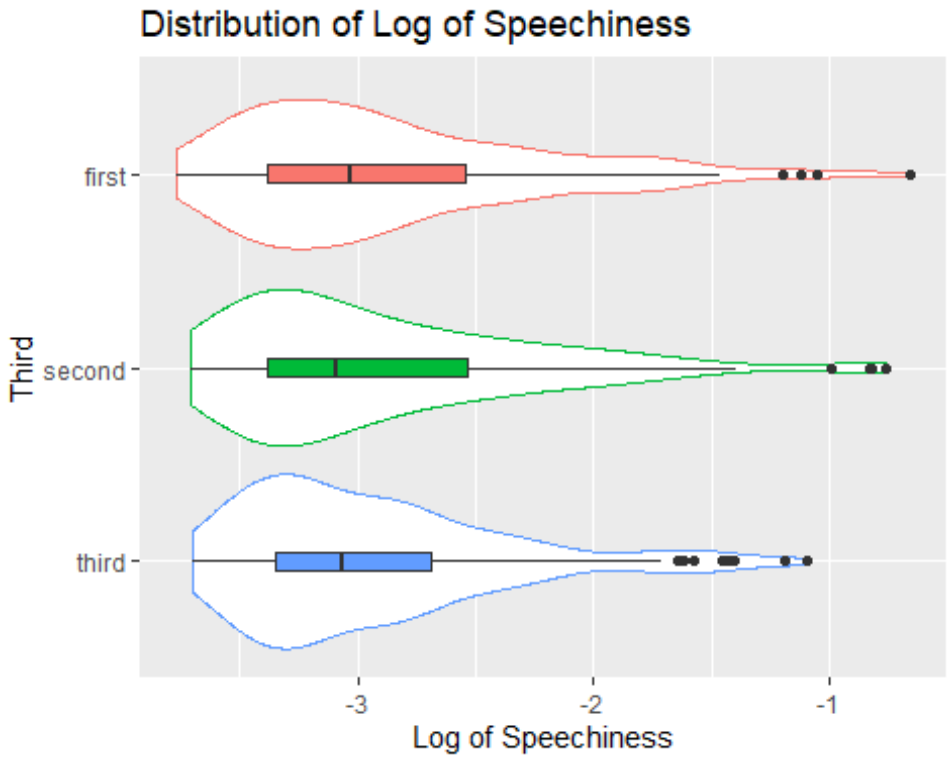

```
# Acousticness by third after 2015 - interesting, got MORE acoustic
songs %>% filter(album_release_year > 2015) %>%
  ggplot(aes(x = factor(song_third, level = c('third', 'second', 'first')),
acousticness)) +
  geom_violin(aes(color = song_third)) +
  geom_boxplot(aes(fill = song_third), width = 0.1) +
  guides(color = F, fill = F) +     # removes legends for color and fill
aesthetics
  labs (title = "Distribution of Acousticness After 2015",
        y = "Acousticness",
        x = "Third") +
  coord_flip()
```



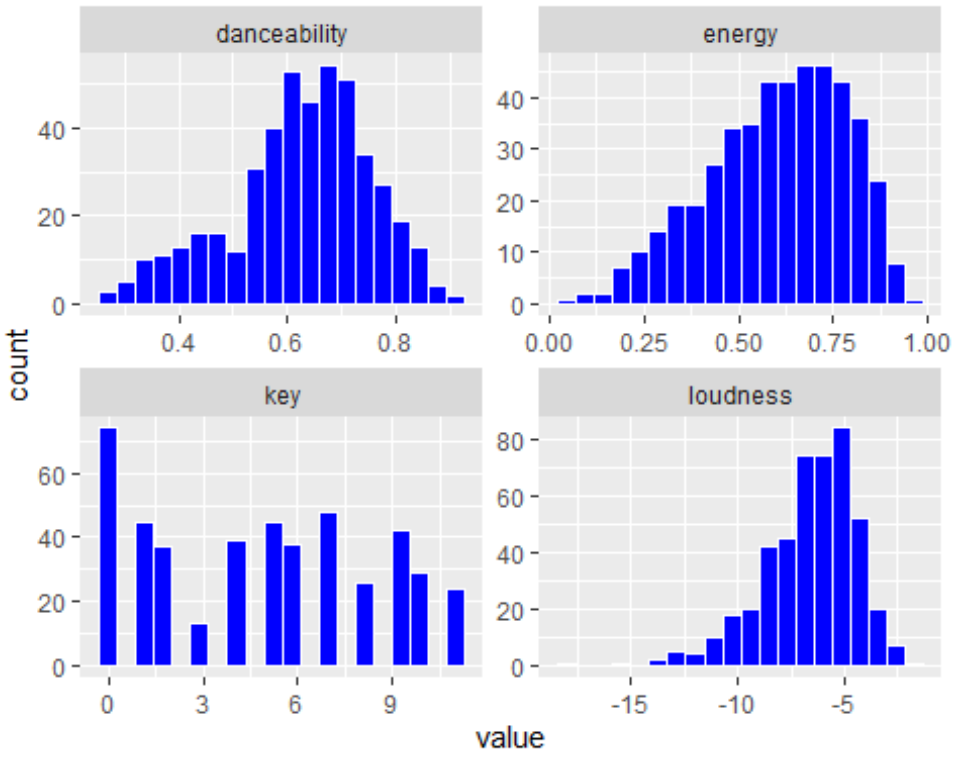

```
# Speechiness by third
ggplot(songs, aes(x = factor(song_third, level = c('third', 'second',
'first')), speechiness)) +
  geom_violin(aes(color = song_third)) +
  geom_boxplot(aes(fill = song_third), width = 0.1) +
  guides(color = F, fill = F) +     # removes legends for color and fill
aesthetics
  labs (title = "Distribution of Speechiness",
        y = "Speechiness",
        x = "Third") +
  coord_flip()
```



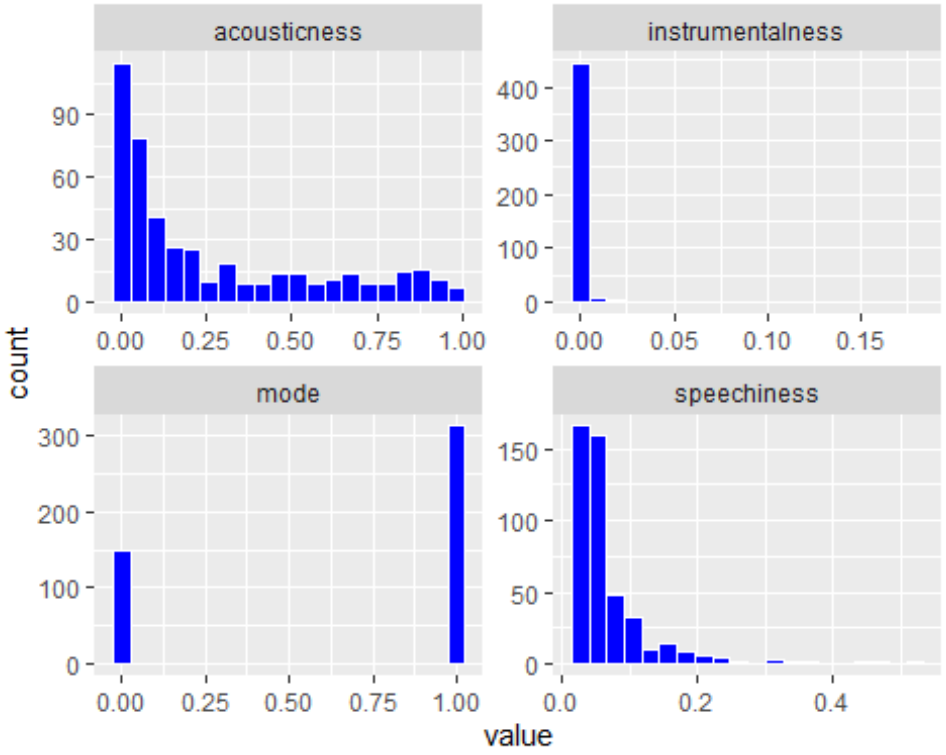

```
# Log Speechiness by third - pretty equal
ggplot(songs, aes(x = factor(song_third, level = c('third', 'second',
'first')), log(speechiness))) +
  geom_violin(aes(color = song_third)) +
  geom_boxplot(aes(fill = song_third), width = 0.1) +
  guides(color = F, fill = F) +     # removes legends for color and fill
aesthetics
  labs (title = "Distribution of Log of Speechiness",
        y = "Log of Speechiness",
        x = "Third") +
  coord_flip()
```



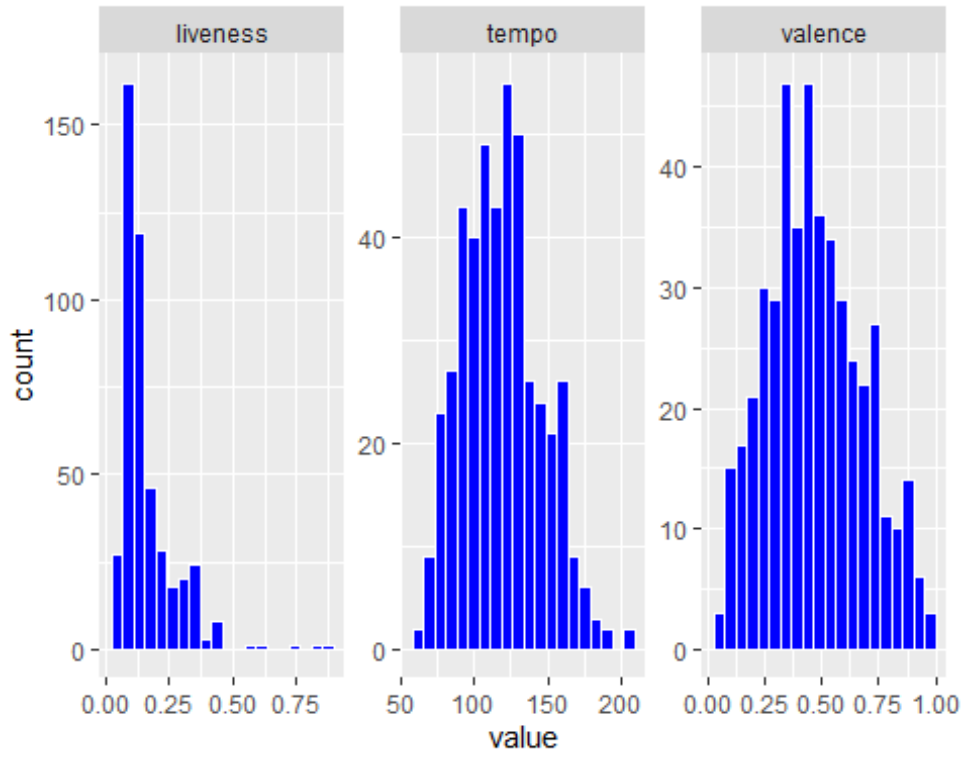

```
# variable plots
songs_num %>%
  pivot_longer(1:4,"varname","value") %>%
  ggplot() +
  geom_histogram(aes(value),fill="blue", color="white", bins = 20) +
  facet_wrap(~ varname, scales = "free")
```



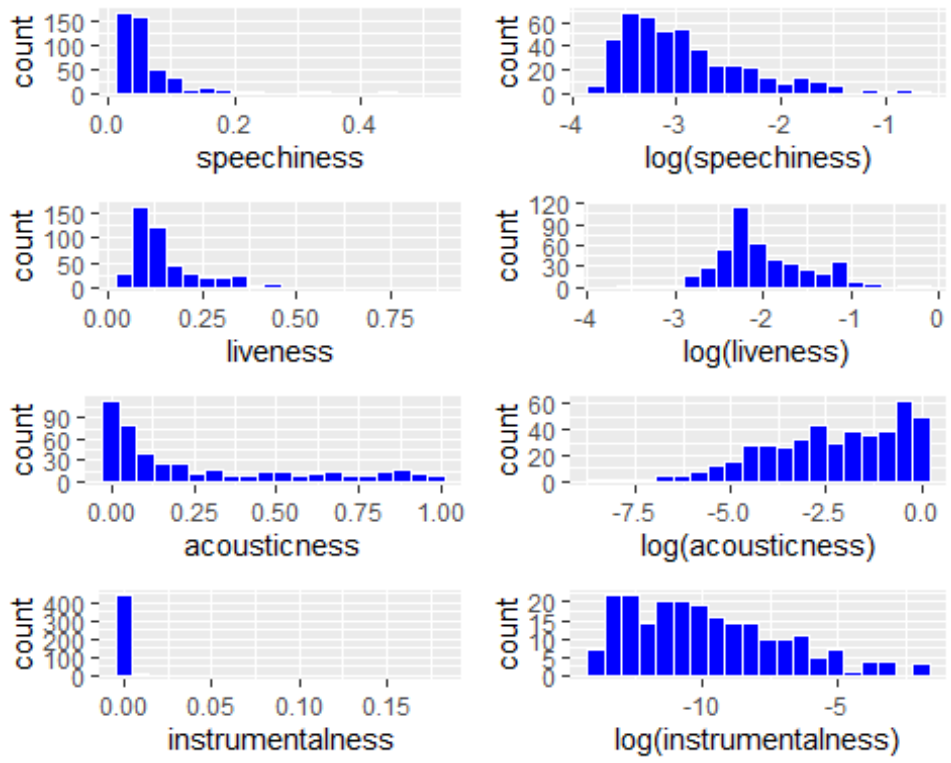

```
songs_num %>% # instrumentalness no good
  pivot_longer(5:8,"varname","value") %>%
  ggplot() +
  geom_histogram(aes(value),fill="blue", color="white", bins = 20) +
  facet_wrap(~ varname, scales = "free")
```



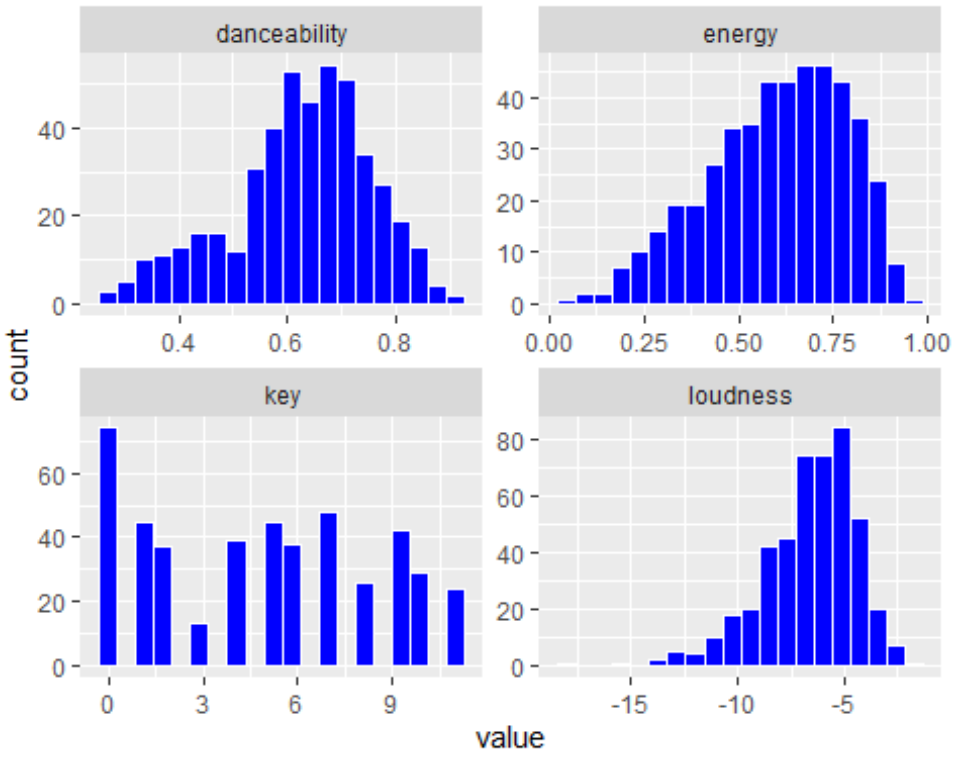

```
songs_num %>%  # time signature isn't good
  pivot_longer(9:11,"varname","value") %>%
  ggplot() +
  geom_histogram(aes(value),fill="blue", color="white", bins = 20) +
  facet_wrap(~ varname, scales = "free")
```



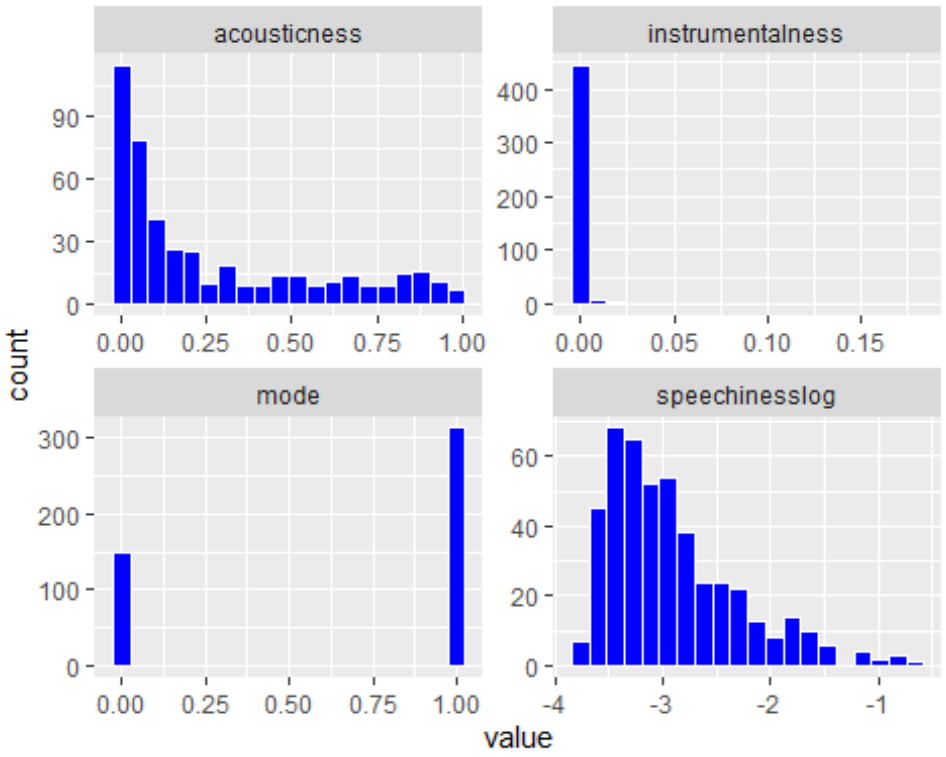

```
# log speechiness, liveliness and acousticness
sp<-songs %>% ggplot() + geom_histogram(aes(speechiness),fill="blue",
color="white", bins = 20)
lsp<-songs %>% ggplot() + geom_histogram(aes(log(speechiness)),fill="blue",
color="white", bins = 20)
li<-songs %>% ggplot() + geom_histogram(aes(liveness), fill="blue",
color="white", bins = 20)
lli<-songs %>% ggplot() + geom_histogram(aes(log(liveness)), fill="blue",
color="white", bins = 20)
ac<-songs %>% ggplot() + geom_histogram(aes(acousticness), fill="blue",
color="white", bins = 20)
lac<-songs %>% ggplot() + geom_histogram(aes(log(acousticness)), fill="blue",
color="white", bins = 20) # regular acoustcness is better
ins<-songs %>% ggplot() + geom_histogram(aes(instrumentalness), fill="blue",
color="white", bins = 20)
lins<-songs %>% ggplot() + geom_histogram(aes(log(instrumentalness)),
fill="blue", color="white", bins = 20) # regular acoustcness is better
grid.arrange(sp, lsp,li,lli,ac,lac,ins,lins, ncol=2)
```



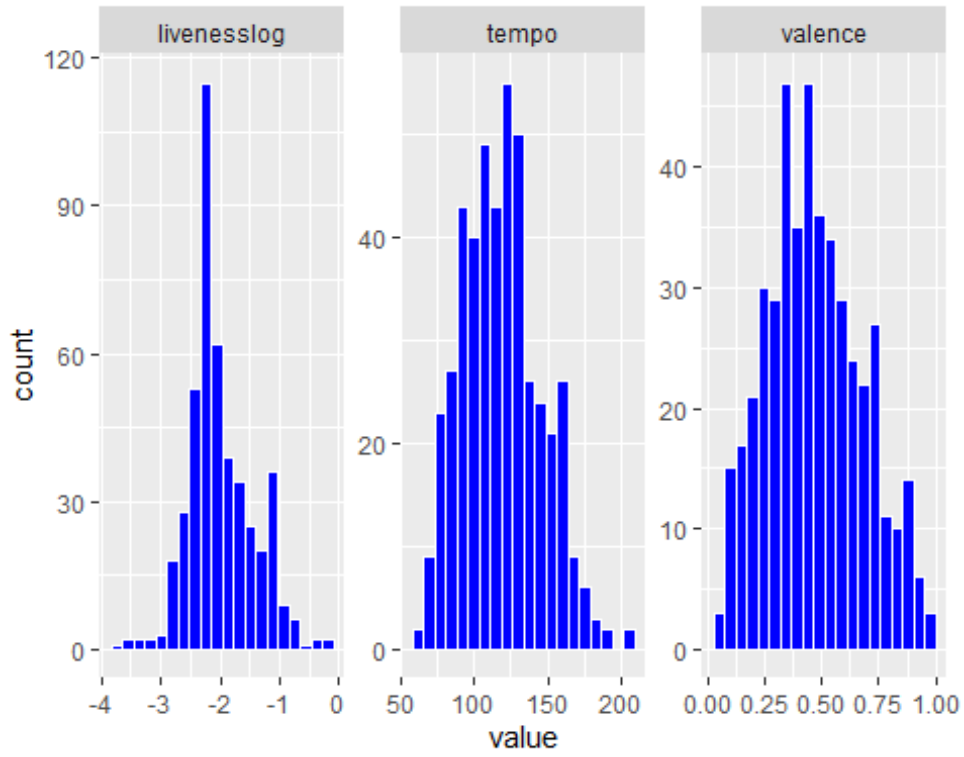

```
# variable plots with logged variables
songs_numlog %>%
  pivot_longer(1:4,"varname","value") %>%
  ggplot() +
  geom_histogram(aes(value),fill="blue", color="white", bins = 20) +
  facet_wrap(~ varname, scales = "free")
```



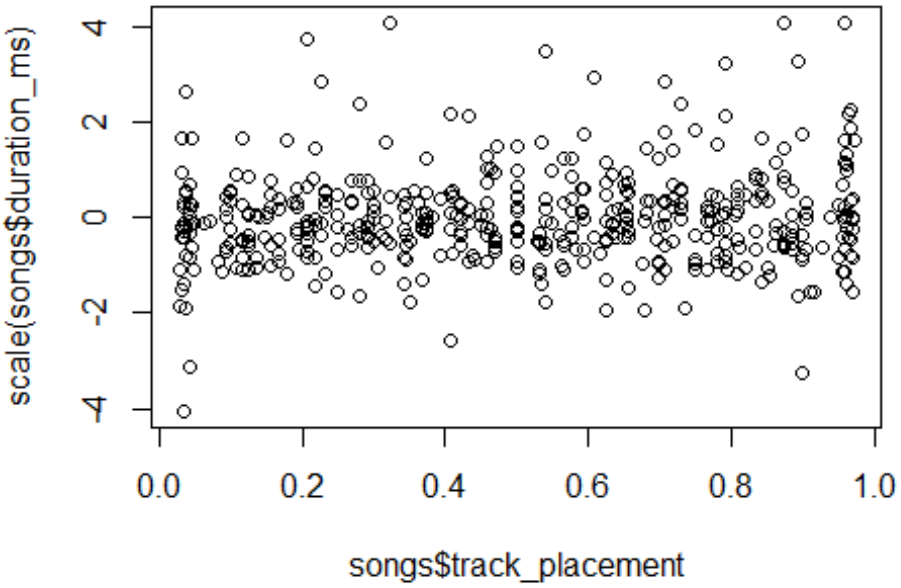

```
songs_numlog %>%
  pivot_longer(5:8,"varname","value") %>%
  ggplot() +
  geom_histogram(aes(value),fill="blue", color="white", bins = 20) +
  facet_wrap(~ varname, scales = "free")
```




```
songs_numlog %>%
  pivot_longer(9:11,"varname","value") %>%
  ggplot() +
  geom_histogram(aes(value),fill="blue", color="white", bins = 20) +
  facet_wrap(~ varname, scales = "free")
```




```r
# pretty even, maybe gets a bit longer
plot(songs$track_placement, scale(songs$duration_ms))
```



```
# Different because of cutoff
table(songs$song_third_num)

##
##   1   2   3
## 144 151 165

# Does explicit change over course of album - not a ton (p=.24)
chisq.test(songs$explicit, songs$song_third) #  %>% prop.table(1)

##
##  Pearson's Chi-squared test
##
## data:  songs$explicit and songs$song_third
## X-squared = 2.8028, df = 2, p-value = 0.2463

# I test for difference in speechiness in first third and others - not sig
(p=.52).
t.test((songs2 %>% filter(song_third=="first") %>%
dplyr::select(speechiness)),
       (songs2 %>% filter(song_third!="first") %>%
dplyr::select(speechiness)))

##
##  Welch Two Sample t-test
##
## data:  (songs2 %>% filter(song_third == "first") %>%
dplyr::select(speechiness)) and (songs2 %>% filter(song_third != "first") %>%
dplyr::select(speechiness))
## t = 0.64632, df = 258.04, p-value = 0.5186
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.008842954  0.017483724
## sample estimates:
##  mean of x  mean of y
## 0.07187292 0.06755253

# I test for difference in log(speechiness) in first third and others - not
sig (p=.47).
t.test((songs2 %>% filter(song_third=="first") %>%
dplyr::select(speechiness)) %>% mutate(speechiness=log(speechiness)),
       (songs2 %>% filter(song_third!="first") %>%
dplyr::select(speechiness))%>% mutate(speechiness=log(speechiness)))

##
##  Welch Two Sample t-test
##
## data:  (songs2 %>% filter(song_third == "first") %>%
dplyr::select(speechiness)) %>% mutate(speechiness = log(speechiness)) and
(songs2 %>% filter(song_third != "first") %>% dplyr::select(speechiness)) %>%
mutate(speechiness = log(speechiness))
```

```
## t = 0.72207, df = 265.3, p-value = 0.4709
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.07792726  0.16818278
## sample estimates:
## mean of x mean of y
## -2.878172 -2.923300
```

```
# I test for difference in speechiness in second third and others - not sig
(p=.63).
t.test((songs2 %>% filter(song_third=="second") %>%
dplyr::select(speechiness)),
       (songs2 %>% filter(song_third!="second") %>%
dplyr::select(speechiness)))
```

```
##
##  Welch Two Sample t-test
##
## data:  (songs2 %>% filter(song_third == "second") %>%
dplyr::select(speechiness)) and (songs2 %>% filter(song_third != "second")
%>% dplyr::select(speechiness))
## t = 0.46955, df = 252.31, p-value = 0.6391
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.01027088  0.01670165
## sample estimates:
##  mean of x  mean of y
## 0.07106490 0.06784951
```

```
# I test for diff in means between third in speechiness - not sig (p = .217)
t.test((songs2 %>% filter(third==1) %>% dplyr::select(speechiness)),
       (songs2 %>% filter(third==0) %>% dplyr::select(speechiness)))
```

```
##
##  Welch Two Sample t-test
##
## data:  (songs2 %>% filter(third == 1) %>% dplyr::select(speechiness)) and
(songs2 %>% filter(third == 0) %>% dplyr::select(speechiness))
## t = -1.2355, df = 424.38, p-value = 0.2173
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.018449865  0.004207585
## sample estimates:
##  mean of x  mean of y
## 0.06433818 0.07145932
```

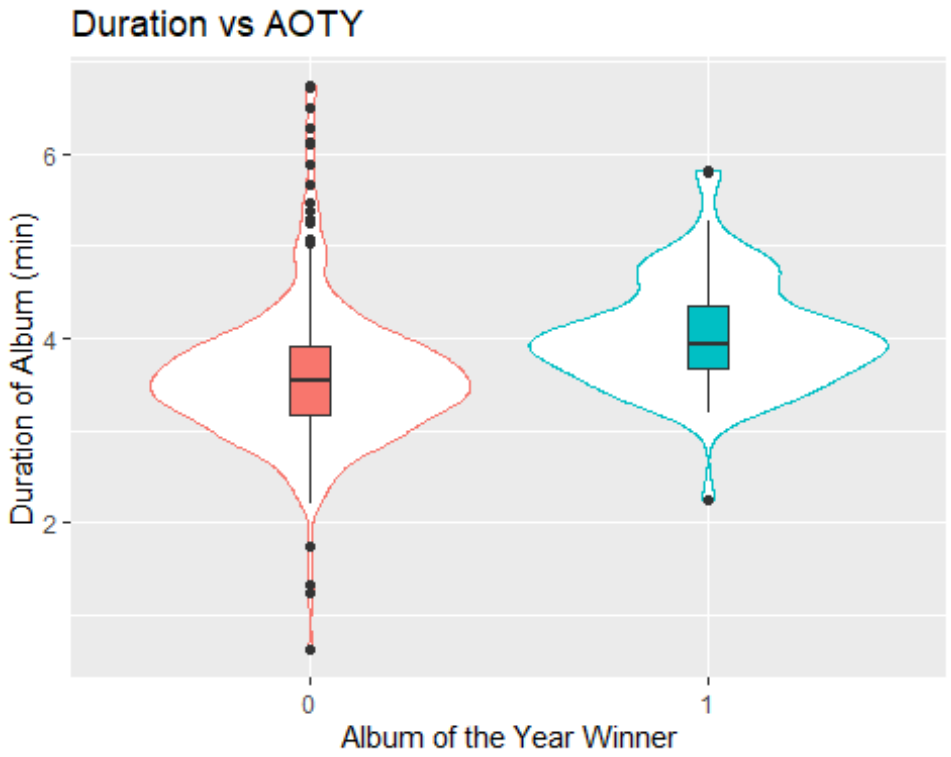
```
# Open energetic, get even more, then mellow in the third
songs %>% group_by(song_third) %>% summarise(mean(energy))
```

```
## # A tibble: 3 x 2
##   song_third `mean(energy)`
```

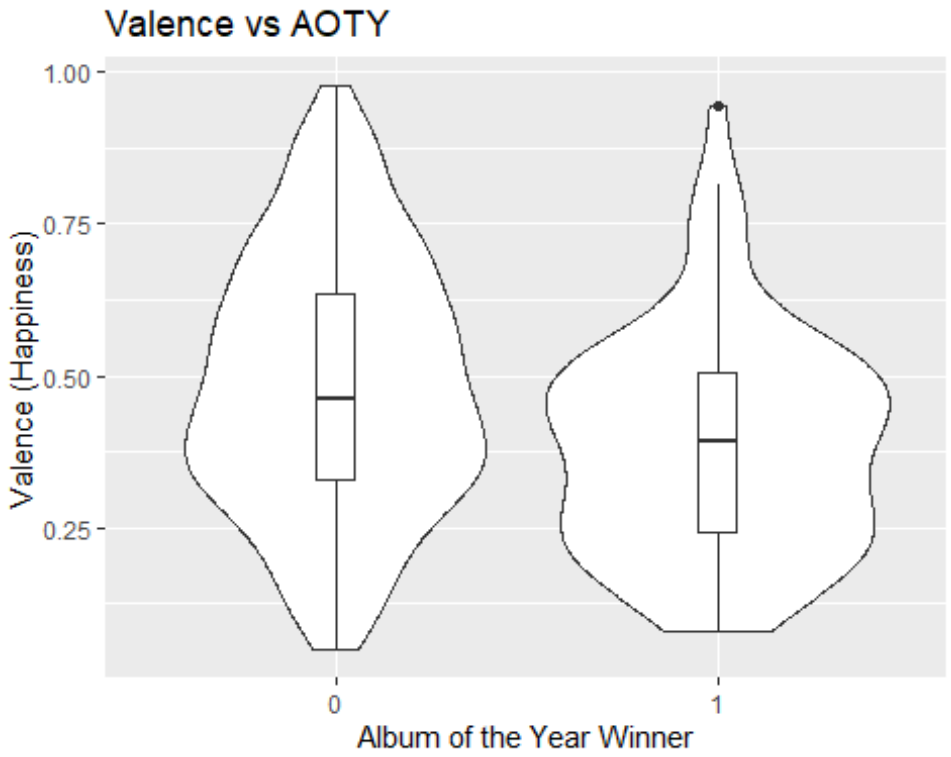
```
##   <chr>               <dbl>
## 1 first               0.602
## 2 second              0.620
## 3 third               0.579
```

Now more exploration of how variables relate to each other

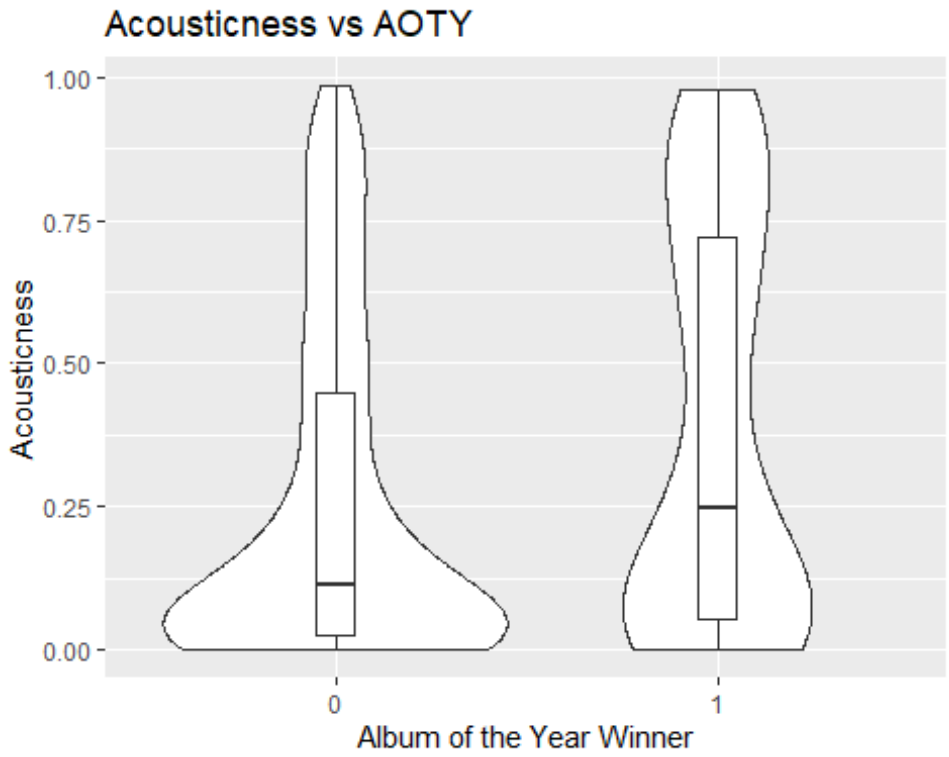
```
# select variables for correlation matrix
# took out: key, mode, explicit, instrumentalist
corr_var <- songs %>%
dplyr::select(c("track_placement","danceability","energy","loudness",
                                        "speechiness", "acousticness",

"liveness","valence","tempo","duration_min"))
# This has logged versions
corr_var2 <- songs %>%
dplyr::select(c("track_placement","danceability","energy","loudness",
                                        "speechiness", "acousticness",

"liveness","valence","tempo","duration_min")) %>%
  mutate(speechiness = log(speechiness),
         liveness = log(liveness))

#corrplot(corrplot.mixed(cor(corr_var)),
 #        lower = "number",
  #       upper = "circle",
   #      tl.col = "black")

# This has logged versions - the differences are very minimal
#corrplot(corrplot.mixed(cor(corr_var2)),
 #        lower = "number",
  #       upper = "circle",
   #      tl.col = "black")



# Distributions by AOTY
songs %>% ggplot() +
  geom_violin(aes(x=as.factor(aoty), y=duration_min, color=as.factor(aoty)))
+
  geom_boxplot(aes(x=as.factor(aoty), y=duration_min,
fill=as.factor(aoty)),width=.1) +
  labs(x="Album of the Year Winner",y="Duration of Album (min)",
       title = "Duration vs AOTY") + theme(legend.position="none")
```



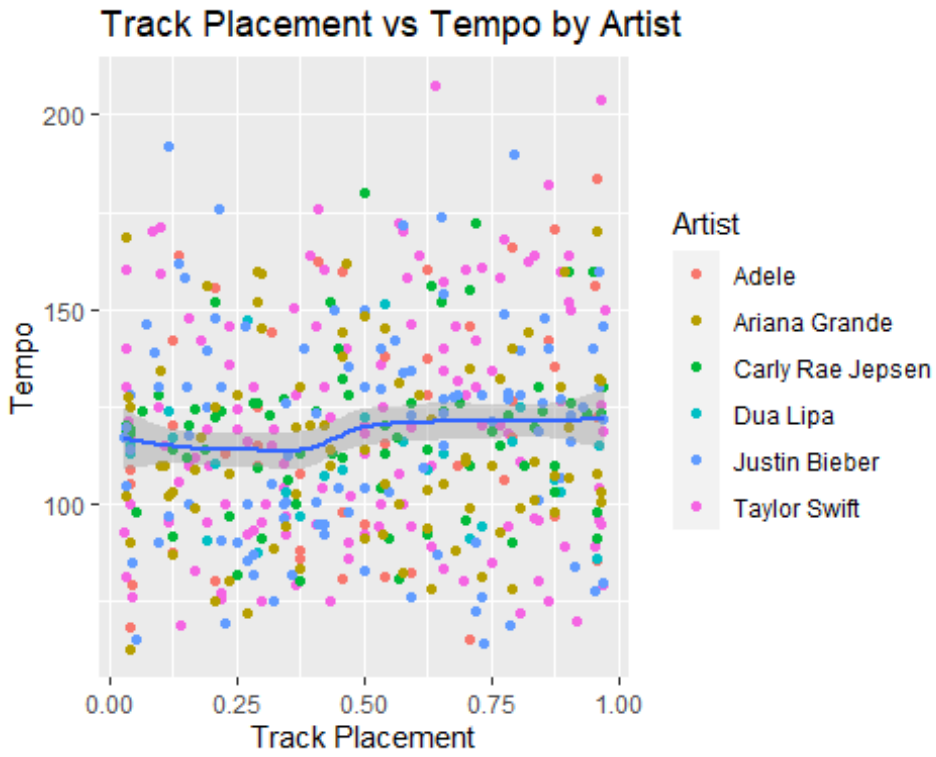

```
songs %>% ggplot() +  geom_violin(aes(x=as.factor(aoty), y=valence)) +
  geom_boxplot(aes(x=as.factor(aoty), y=valence),width=.1) +
  labs(x="Album of the Year Winner",y="Valence (Happiness)", title = "Valence
vs AOTY")
```



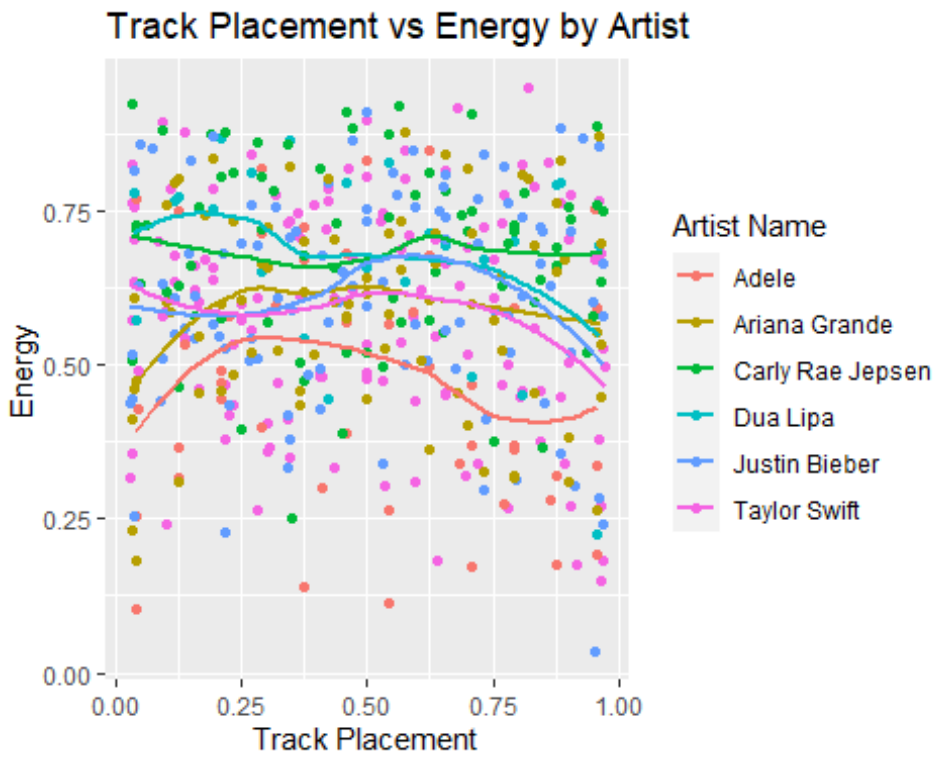

```
songs %>% ggplot() + geom_violin(aes(x=as.factor(aoty), y=acousticness)) +
  geom_boxplot(aes(x=as.factor(aoty), y=acousticness),width=.1) +
  labs(x="Album of the Year Winner",y="Acousticness", title = "Acousticness
vs AOTY")
```



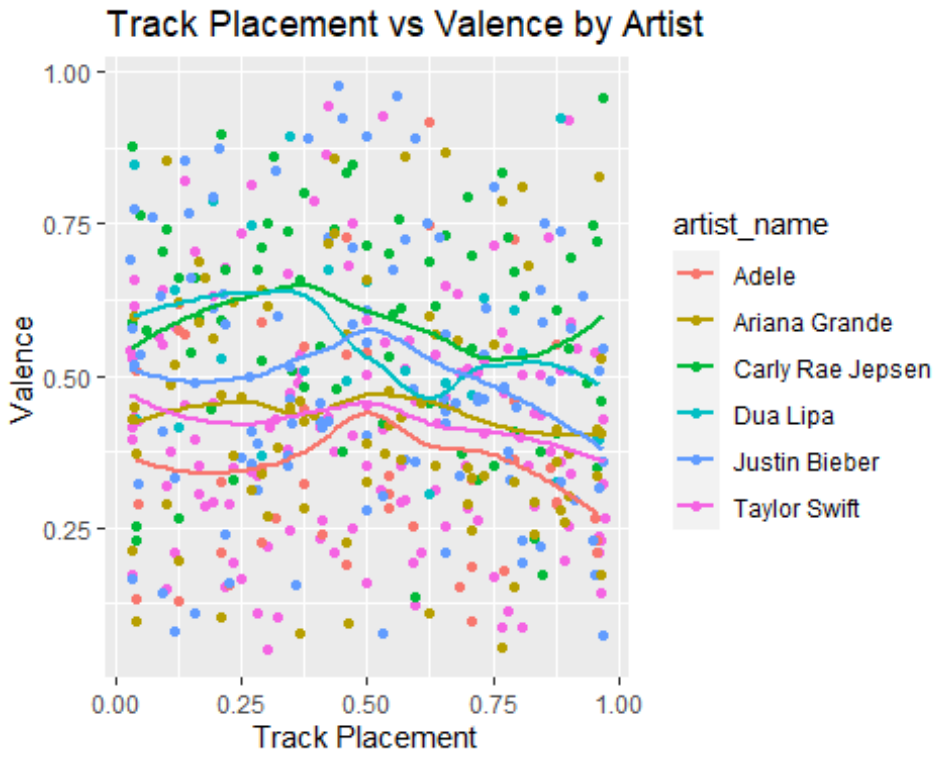

```
# track_placement plots
songs %>% ggplot() + geom_point(aes(x=track_placement, y=tempo,
col=artist_name)) +
  #geom_smooth(aes(x=track_placement, y=tempo,, group=artist_name,
col=artist_name),se=F)+
  geom_smooth(aes(x=track_placement, y=tempo))+
  labs(x="Track Placement", y="Tempo",title="Track Placement vs Tempo by
Artist", color="Artist")
```



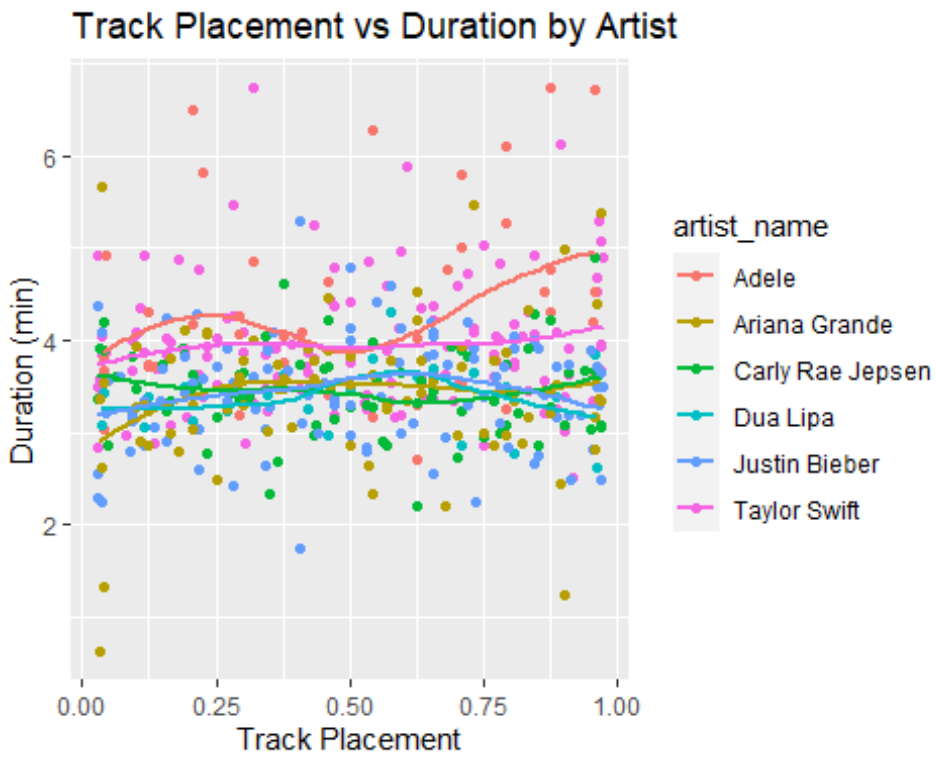

```
songs %>% ggplot() + geom_point(aes(x=track_placement, y=energy,
col=artist_name)) +
  geom_smooth(aes(x=track_placement, y=energy, group=artist_name,
col=artist_name), se=FALSE)+
  labs(x="Track Placement", y="Energy",title="Track Placement vs Energy by
Artist", color = "Artist Name")
```



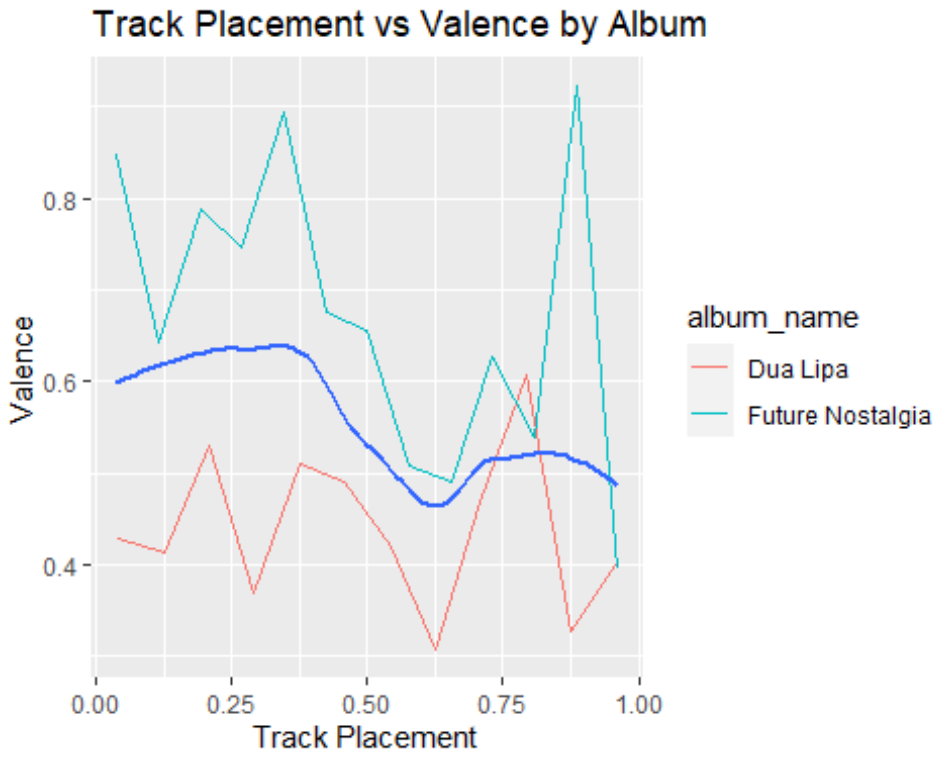

```
songs %>% ggplot() + geom_point(aes(x=track_placement, y=valence,
col=artist_name)) +
  geom_smooth(aes(x=track_placement, y=valence, group=artist_name,
col=artist_name), se=FALSE)+
  labs(x="Track Placement", y="Valence",title="Track Placement vs Valence by
Artist")
```



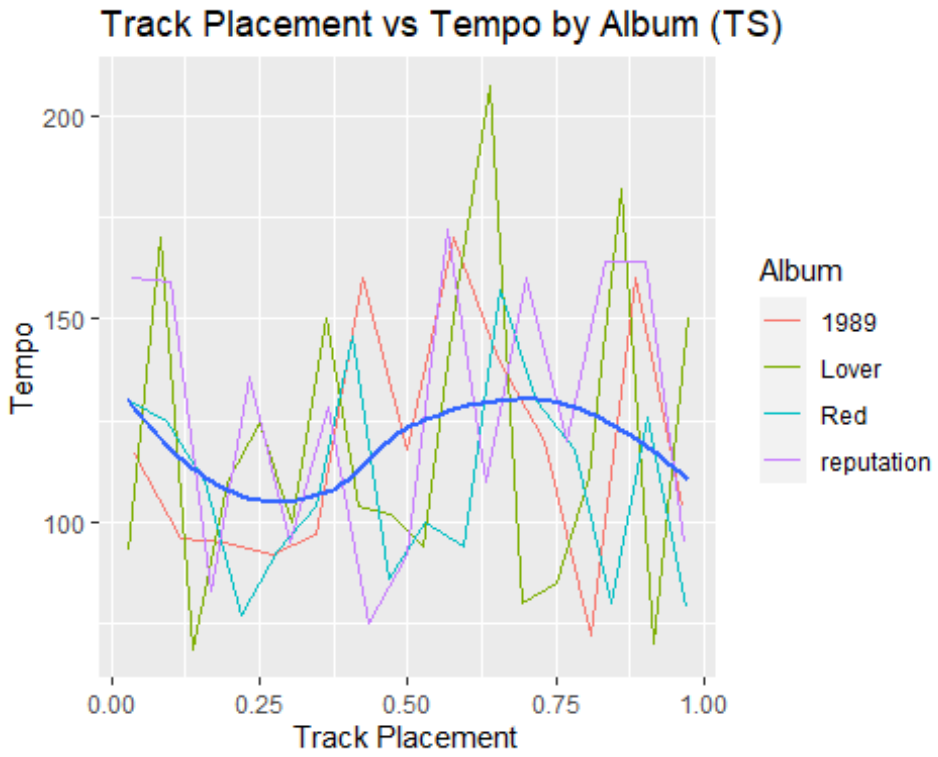

```
songs %>% ggplot() + geom_point(aes(x=track_placement, y=duration_min,
col=artist_name)) +
  geom_smooth(aes(x=track_placement, y=duration_min, group=artist_name,
col=artist_name), se=FALSE)+
  labs(x="Track Placement", y="Duration (min)",title="Track Placement vs
Duration by Artist")
```



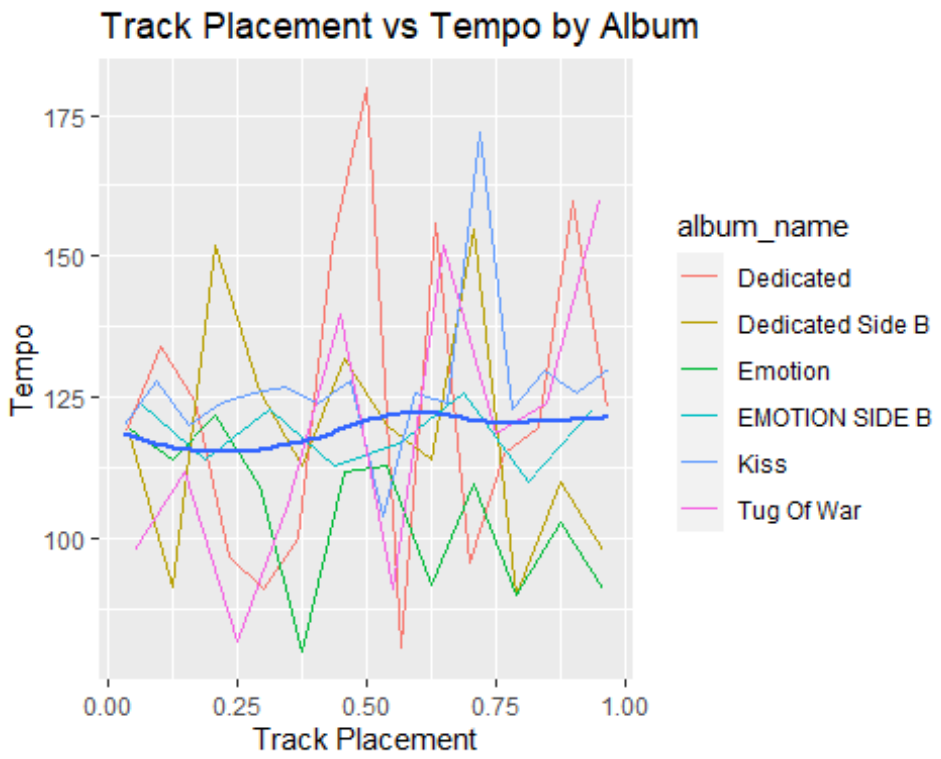

```
# Artist Plots
# Dua Lipa's songs get sadder throughout the album
songs %>% filter(artist_name == "Dua Lipa") %>% ggplot() +
geom_line(aes(x=track_placement, y=valence, col=album_name)) +
  geom_smooth(aes(x=track_placement, y=valence), se=FALSE)+
  labs(x="Track Placement", y="Valence",title="Track Placement vs Valence by
Album")
```



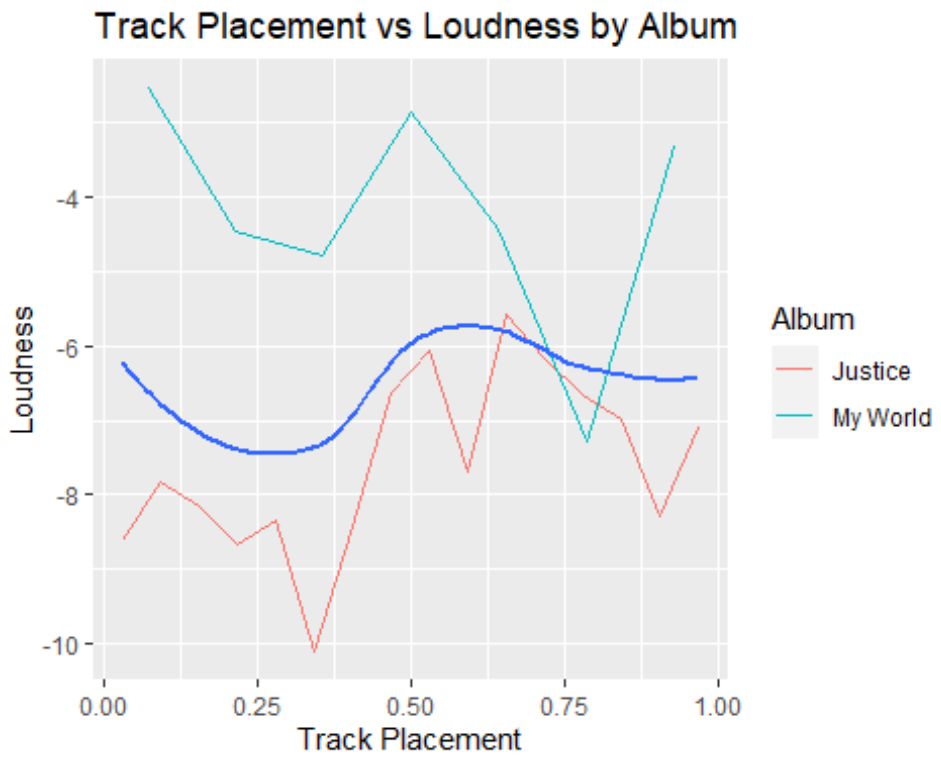

```
# Tempo by Swift's later albums
songs %>% filter(artist_name == "Taylor Swift") %>%
  filter(album_release_year > 2010 & album_release_year<2020) %>%
  ggplot() + geom_line(aes(x=track_placement, y=tempo, col=album_name)) +
  geom_smooth(aes(x=track_placement, y=tempo), se=FALSE)+
  labs(x="Track Placement", y="Tempo",title="Track Placement vs Tempo by
Album (TS)", color="Album")
```



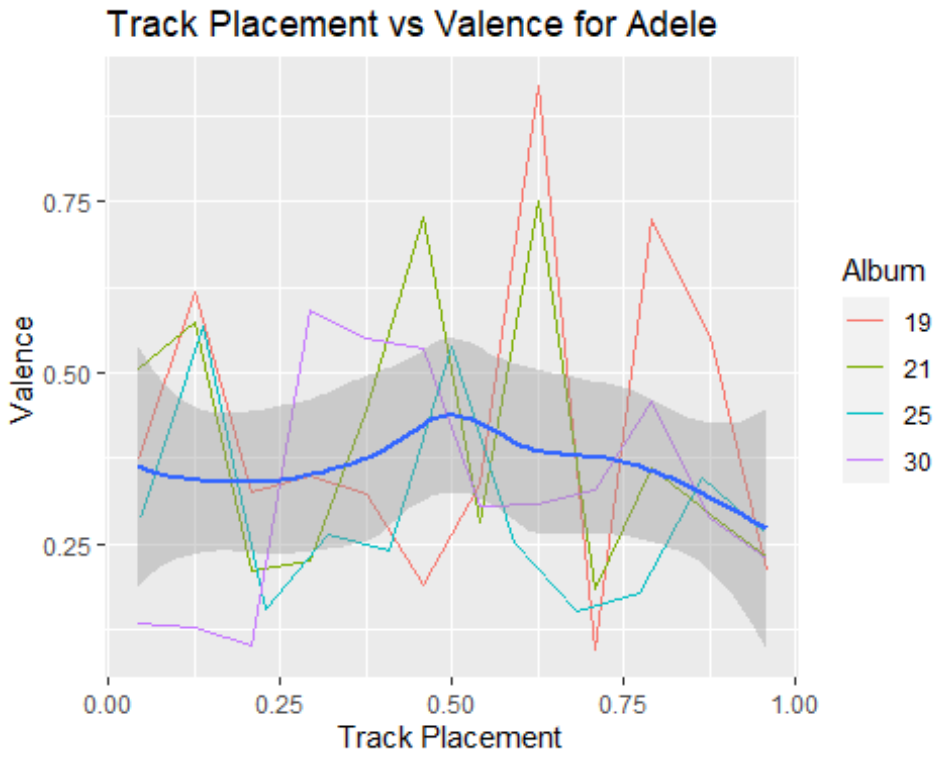

```
songs %>% filter(artist_name == "Carly Rae Jepsen") %>% ggplot() +
geom_line(aes(x=track_placement, y=tempo, col=album_name)) +
  geom_smooth(aes(x=track_placement, y=tempo), se=FALSE)+
  labs(x="Track Placement", y="Tempo",title="Track Placement vs Tempo by
Album")
```



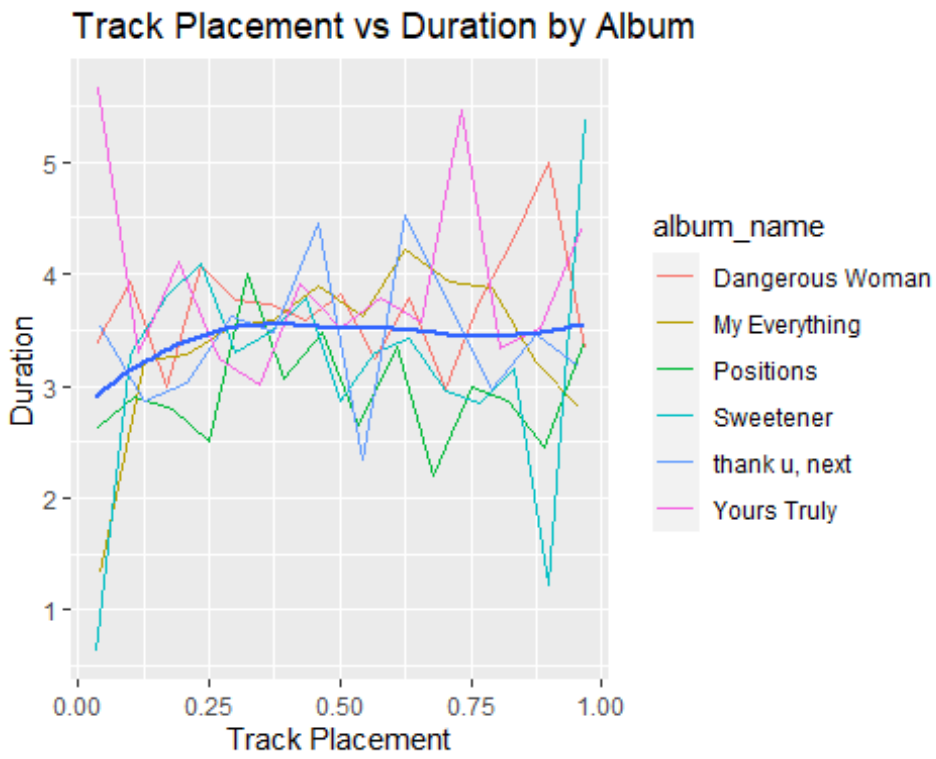

```
songs %>% filter(artist_name == "Justin Bieber") %>%
  filter(album_name=="My World" | album_name == "Justice") %>%
  ggplot() + geom_line(aes(x=track_placement, y=loudness, col=album_name)) +
  geom_smooth(aes(x=track_placement, y=loudness), se=FALSE)+
  labs(x="Track Placement", y="Loudness",title="Track Placement vs Loudness
by Album", color = "Album")
```



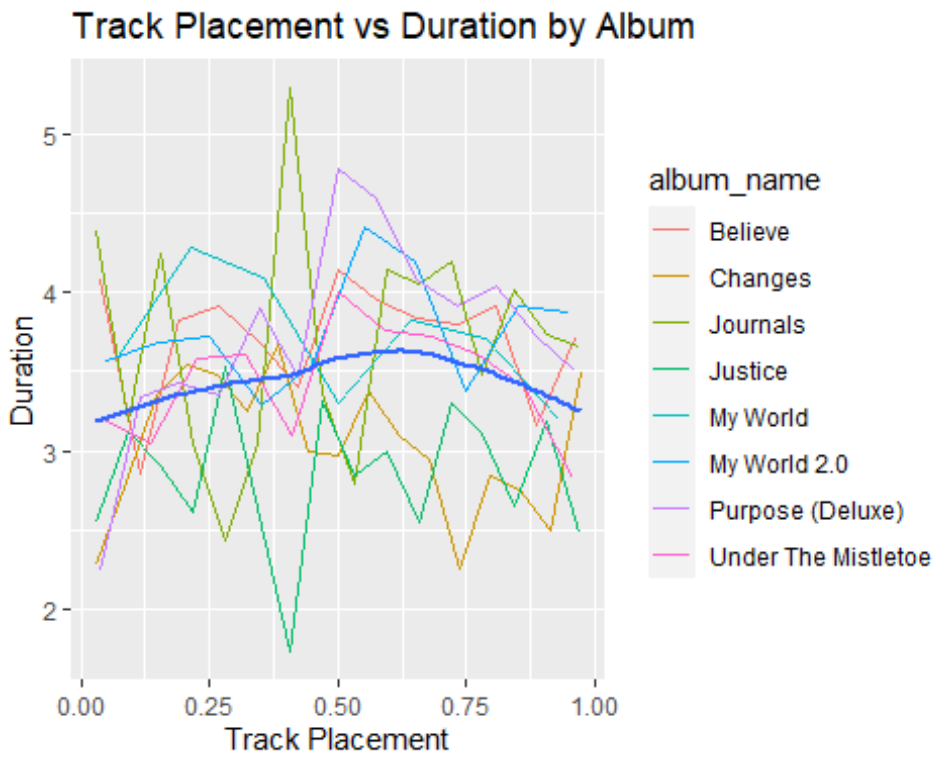

```
# High happiness songs followed by low ones
songs %>% filter(artist_name == "Adele") %>% ggplot() +
geom_line(aes(x=track_placement, y=valence, col=album_name)) +
  geom_smooth(aes(x=track_placement, y=valence))+
  labs(x="Track Placement", y="Valence",title="Track Placement vs Valence for
Adele",
       color = "Album")
```



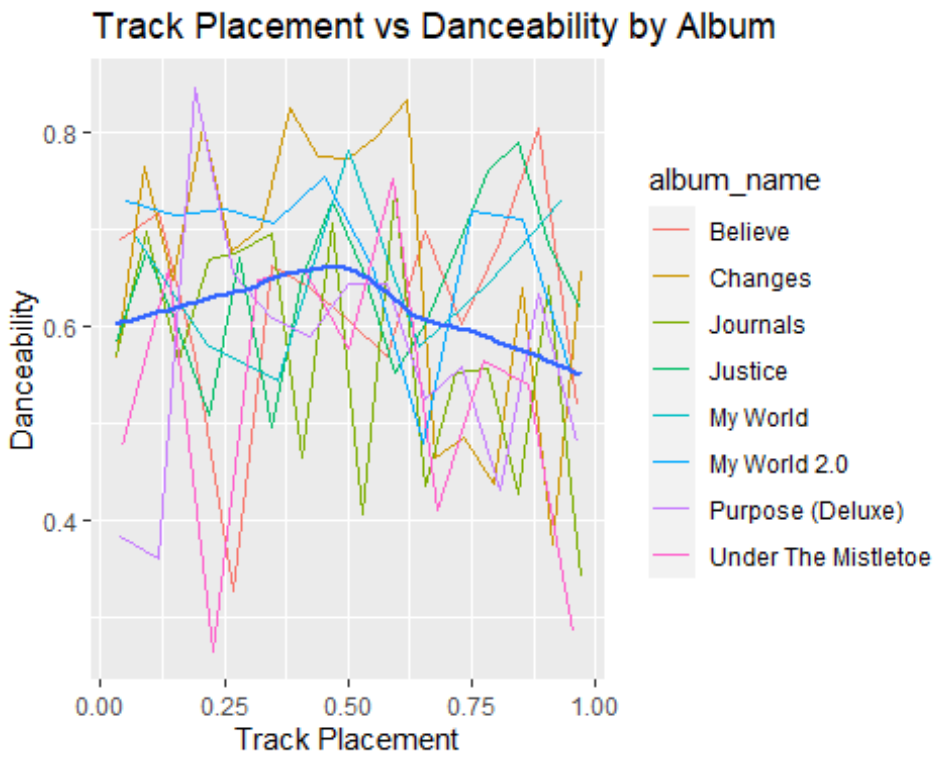

```
# songs in the middle of Adele's albums are short
# Ariana likes to start with an outlier
songs %>% filter(artist_name == "Ariana Grande") %>% ggplot() +
geom_line(aes(x=track_placement, y=duration_min, col=album_name)) +
  geom_smooth(aes(x=track_placement, y=duration_min), se=FALSE)+
  labs(x="Track Placement", y="Duration",title="Track Placement vs Duration
by Album")
```



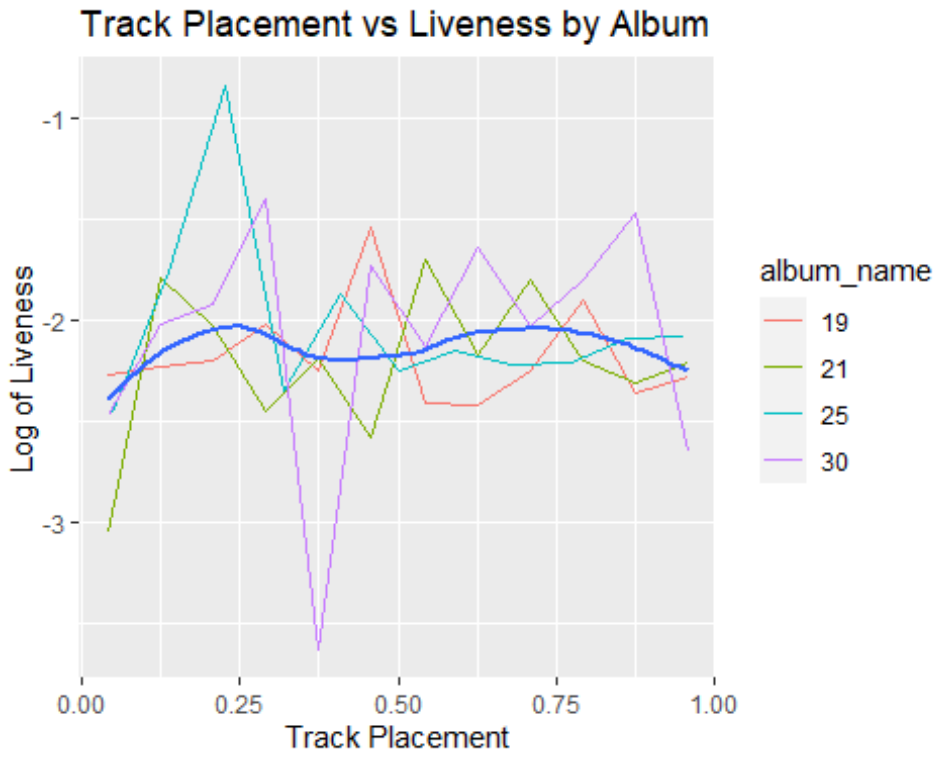

```
# songs in middle of Justin are longer
songs %>% filter(artist_name == "Justin Bieber") %>% ggplot() +
geom_line(aes(x=track_placement, y=duration_min, col=album_name)) +
  geom_smooth(aes(x=track_placement, y=duration_min), se=FALSE)+
  labs(x="Track Placement", y="Duration",title="Track Placement vs Duration
by Album")
```




```
# Very up and down danceability for Adele
# Justin goes up the first half of the ablum and down the second half
songs %>% filter(artist_name == "Justin Bieber") %>% ggplot() +
geom_line(aes(x=track_placement, y=danceability, col=album_name)) +
  geom_smooth(aes(x=track_placement, y=danceability), se=FALSE)+
  labs(x="Track Placement", y="Danceability",title="Track Placement vs
Danceability by Album")
```




```
# Liveness - bottoms at beginning of second third
songs %>% filter(artist_name == "Adele") %>% ggplot() +
geom_line(aes(x=track_placement, y=log(liveness), col=album_name)) +
  geom_smooth(aes(x=track_placement, y=log(liveness)), se=FALSE)+
  labs(x="Track Placement", y="Log of Liveness",title="Track Placement vs
Liveness by Album")
```



Supervised Learning - Classification

Here is QDA

```
######################
# QDA
######################
mod_qda <- qda(song_third_num ~ danceability+energy+key+loudness+mode+
                 speechiness+acousticness+instrumentalness+liveness+
                 valence+tempo+duration_min+explicit+aoty+
                 album_length, data = songs )

songs %>% # 49.78 so not great but I've seen worse
  mutate(pred_third = predict(mod_qda, newdata = songs)$class) %>%
  dplyr::select(song_third_num, pred_third) %>%
  table()

##                pred_third
## song_third_num   1   2   3
##              1  38  89  17
##              2   8 126  17
##              3  15  85  65

######################
# QDA with logged vars
######################
```

```
mod_qda_log <- qda(song_third_num ~ danceability+energy+key+loudness+mode+
                  speechinesslog+acousticness+instrumentalness+livenesslog+
                  valence+tempo+duration_min+explicit+aoty+
                  album_length, data = songs_log )

songs %>% # 51.30 so a bit better
  mutate(pred_third = predict(mod_qda_log, newdata = songs_log)$class) %>%
  dplyr::select(song_third_num, pred_third) %>%
  table()

##                pred_third
## song_third_num   1   2   3
##              1  40  89  15
##              2   5 133  13
##              3  16  86  63
```

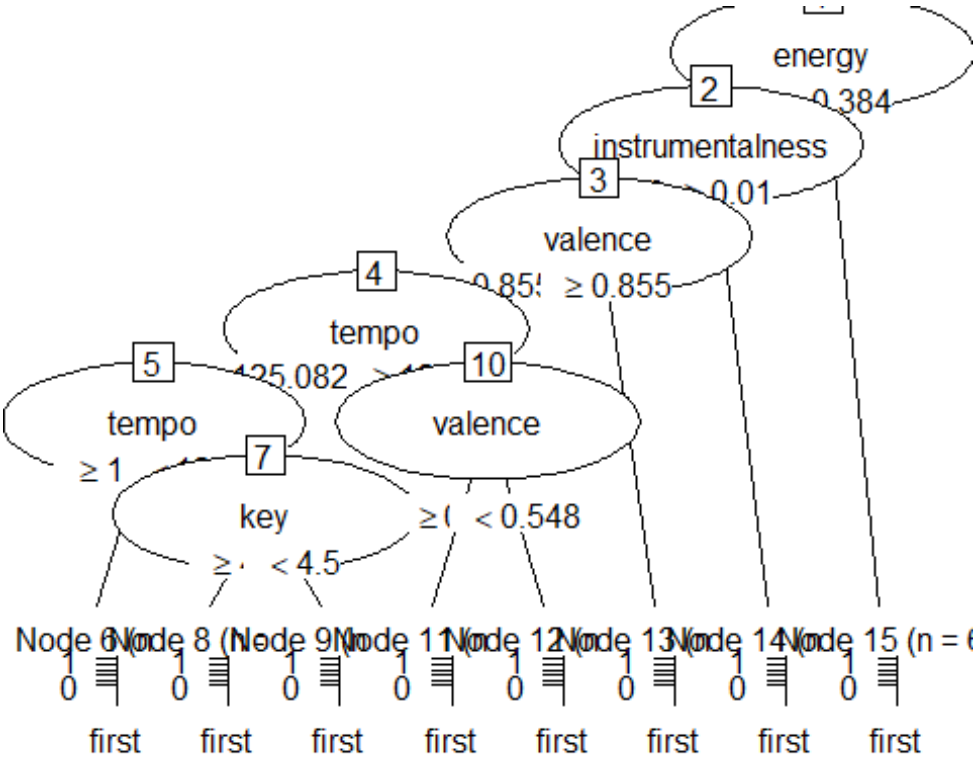
Here is Naive Bayes

```
######################
# Naïve Bayes
######################

nb <- naiveBayes(song_third_num ~ danceability+energy+key+loudness+mode+
                   speechiness+acousticness+instrumentalness+liveness+
                   valence+tempo+duration_min+explicit+aoty+
                   album_length, data = songs )

songs %>% # 37.61% so less than QDA and bad at predicting first third
  mutate(nb_pred = predict(nb, newdata = songs),
         match = (song_third_num == nb_pred)) %>%
  dplyr::select(song_third_num,nb_pred) %>%
  table()

##                nb_pred
## song_third_num   1   2   3
##              1   0 125  19
##              2   1 127  23
##              3   3 116  46

######################
# Naive Bayes with logged vars
######################

nb_log <- naiveBayes(song_third_num ~ danceability+energy+key+loudness+mode+
                   speechinesslog+acousticness+instrumentalness+livenesslog+
                   valence+tempo+duration_min+explicit+aoty+
                   album_length, data = songs_log )

songs_log %>% # 38.47% so not a lot better
  mutate(nb_pred = predict(nb_log, newdata = songs_log),
```

```
         match = (song_third_num == nb_pred)) %>%
  dplyr::select(song_third_num,nb_pred) %>%
  table()

##                nb_pred
## song_third_num   1   2   3
##              1   0 127  17
##              2   0 131  20
##              3   3 116  46
```

Here is using classification trees

```
######################
# Classification
######################

treetop <- rpart::rpart(song_third ~ danceability+energy+key+loudness+mode+
                          speechiness+acousticness+instrumentalness+liveness+

valence+tempo+duration_min+explicit+aoty+album_length,
                        data=songs, cp = .025)
treetop %>% as.party() %>% plot()
```

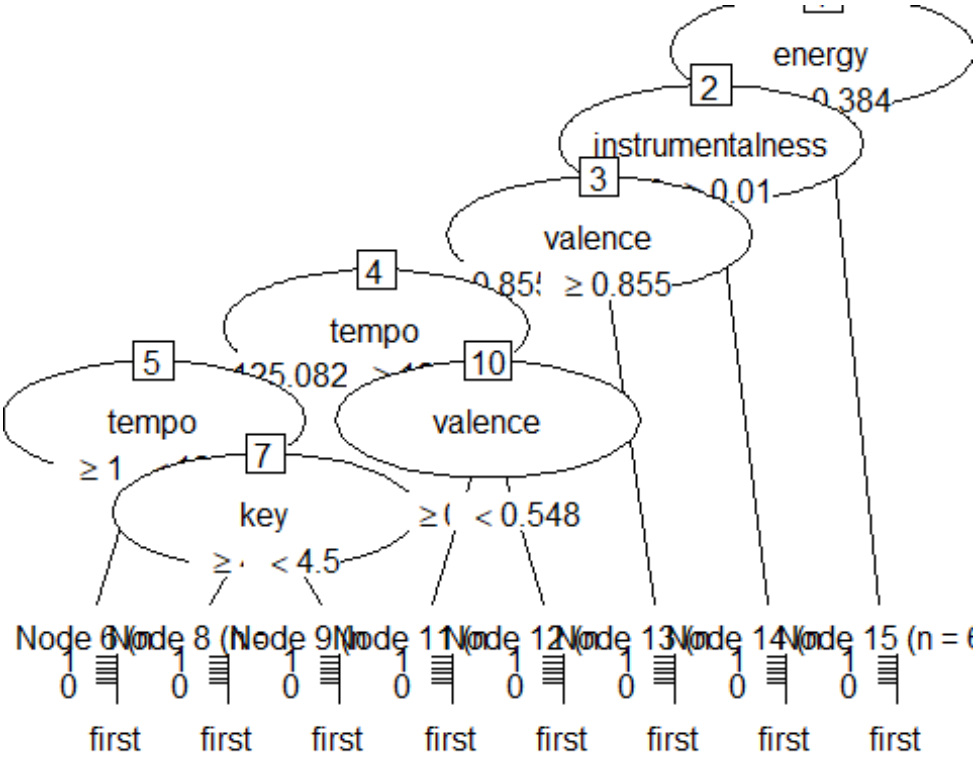



```
# percent right: 51.09% so a little better
data.frame(songs,predict(treetop, newdata=songs)) %>%
  mutate(id = c(1:nrow(songs))) %>% # give each row an id
  group_by(id) %>%
```

```
  mutate(max_prob = max(first, second, third), # create columns for prob of
each cat and the max
         pred = case_when(                              # assigned to cat
with highest prob
           max_prob == first ~ "first",
           max_prob == second ~ "second",
           max_prob == third ~ "third")) %>%
  ungroup() %>% # undo the group by row
  dplyr::select(song_third,pred) %>%
  table()
```

```
##           pred
## song_third first second third
##     first     84     27    33
##     second    51     58    42
##     third     44     28    93
```

```
######################
# Classification with logged variables
######################

treetop_log <- rpart::rpart(song_third ~
danceability+energy+key+loudness+mode+

speechinesslog+acousticness+instrumentalness+livenesslog+

valence+tempo+duration_min+explicit+aoty+album_length,
                        data=songs_log, cp = .025)
treetop_log %>% as.party() %>% plot()
```

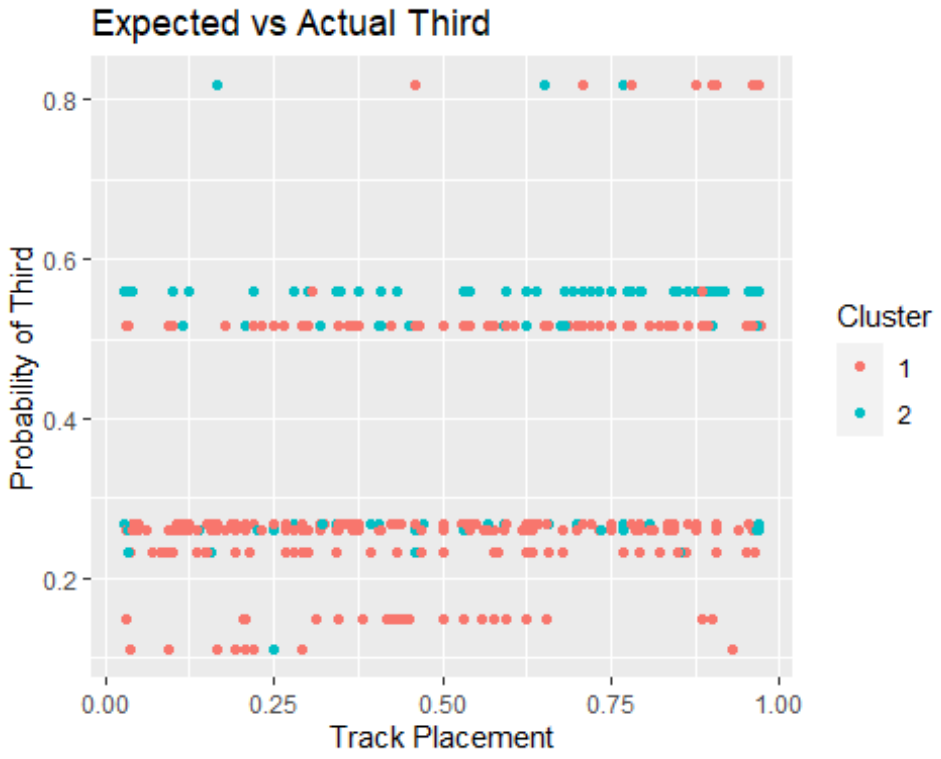



```
# percent right: 51.09% so exactly the same
data.frame(songs_log,predict(treetop_log, newdata=songs_log)) %>%
  mutate(id = c(1:nrow(songs_log))) %>% # give each row an id
  group_by(id) %>%
  mutate(max_prob = max(first, second, third), # create columns for prob of
each cat and the max
         pred = case_when(                              # assigned to cat
with highest prob
           max_prob == first ~ "first",
           max_prob == second ~ "second",
           max_prob == third ~ "third")) %>%
  ungroup() %>% # undo the group by row
  dplyr::select(song_third,pred) %>%
  table()

##           pred
## song_third first second third
##     first     84     27    33
##     second    51     58    42
##     third     44     28    93
```

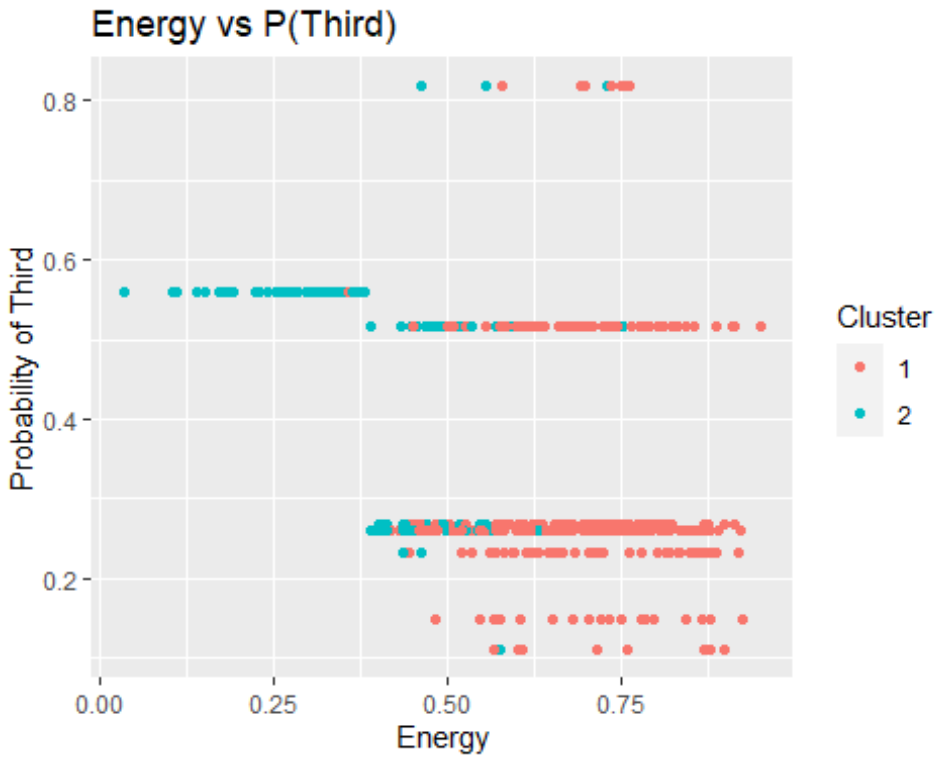
Some plots relevant to the above classification trees

```
# This plots track_placement against probability of being in third part of
album. Colored by cluster.
data.frame(songs,predict(treetop, newdata=songs)) %>%
  mutate(clust = kmeans(scale(songs_num),2)$cluster) %>%
```

```
  ggplot() + geom_point(aes(x=track_placement,y=third, col=as.factor(clust)))
+
  labs(color = "Cluster", x="Track Placement", y= "Probability of Third",
title="Expected vs Actual Third")
```

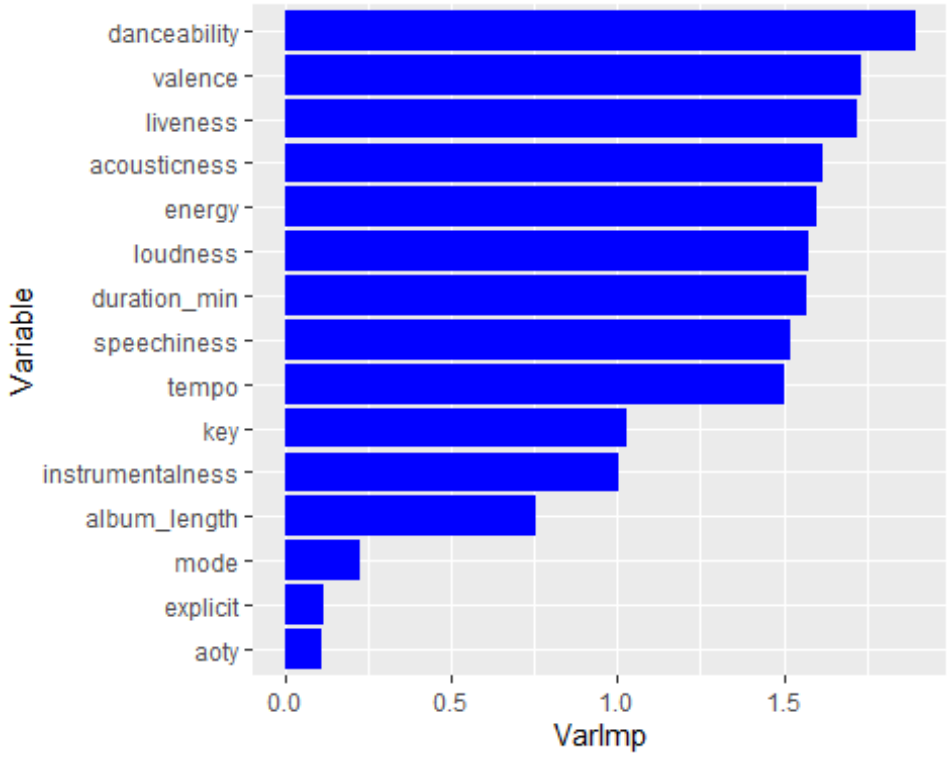



```
data.frame(songs,predict(treetop, newdata=songs)) %>%
  mutate(clust = kmeans(scale(songs_num),2)$cluster) %>%
  ggplot() + geom_point(aes(x=energy,y=third, col=as.factor(clust))) +
  labs(color = "Cluster", x="Energy", y= "Probability of Third",
title="Energy vs P(Third)")
```



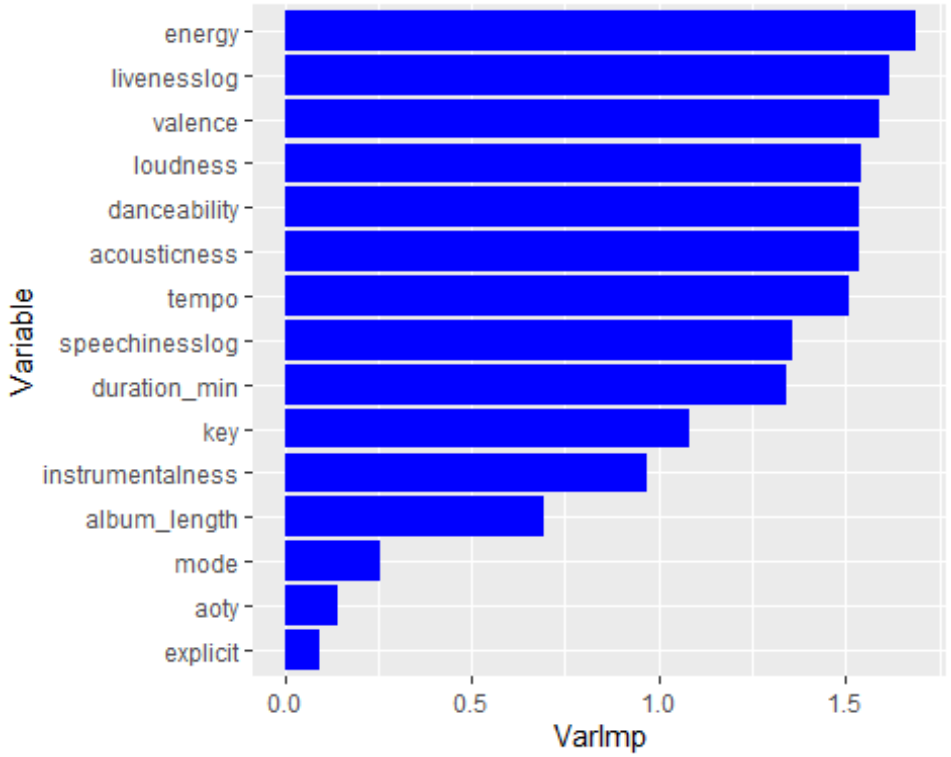
Here is classification using bagging

```
######################
# Bagging
######################

bag1 <- bagging(song_third_num ~ ., data = songs_pred)
vi1 <- varImp(bag1)  # varImp in caret

ggplot() + geom_col(aes(reorder(rownames(vi1),-desc(vi1$Overall)),
vi1$Overall), fill="blue") +
  labs(x = "Variable", y = "VarImp") +
  coord_flip()
```




```
# Always changes - 43.48% Accuracy - it only ever predicted 2nd third of the album
table(round(predict(bag1,newdata=songs)),songs$song_third_num) %>%
confusionMatrix()

## Confusion Matrix and Statistics
##
##
##       1   2   3
##   1  11   0   0
##   2 133 150 116
##   3   0   1  49
##
## Overall Statistics
##
##                Accuracy : 0.4565
##                  95% CI : (0.4103, 0.5033)
##     No Information Rate : 0.3587
##     P-Value [Acc > NIR] : 1.016e-05
##
##                   Kappa : 0.1874
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
```

```
## Sensitivity           0.07639   0.9934   0.2970
## Specificity           1.00000   0.1942   0.9966
## Pos Pred Value        1.00000   0.3759   0.9800
## Neg Pred Value        0.70379   0.9836   0.7171
## Prevalence            0.31304   0.3283   0.3587
## Detection Rate        0.02391   0.3261   0.1065
## Detection Prevalence  0.02391   0.8674   0.1087
## Balanced Accuracy     0.53819   0.5938   0.6468

#######################
# Bagging with logged vars
#######################
bag1log <- bagging(song_third_num ~ ., data = songs_predlog)
vi1log <- varImp(bag1log)  # varImp in caret

ggplot() + geom_col(aes(reorder(rownames(vi1log),-desc(vi1log$Overall)),
vi1log$Overall), fill="blue") +
  labs(x = "Variable", y = "VarImp") +
  coord_flip()
```




```
# 44.13% Accuracy - actually does worse
table(round(predict(bag1log,newdata=songs_log)),songs_log$song_third_num) %>%
confusionMatrix()

## Confusion Matrix and Statistics
##
##
```

```
##       1   2   3
##   1   9   0   0
##   2 135 150 125
##   3   0   1  40
##
## Overall Statistics
##
##                Accuracy : 0.4326
##                  95% CI : (0.3868, 0.4793)
##     No Information Rate : 0.3587
##     P-Value [Acc > NIR] : 0.000642
##
##                   Kappa : 0.1523
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity           0.06250   0.9934  0.24242
## Specificity           1.00000   0.1586  0.99661
## Pos Pred Value        1.00000   0.3659  0.97561
## Neg Pred Value        0.70067   0.9800  0.70167
## Prevalence            0.31304   0.3283  0.35870
## Detection Rate        0.01957   0.3261  0.08696
## Detection Prevalence  0.01957   0.8913  0.08913
## Balanced Accuracy     0.53125   0.5760  0.61952
```

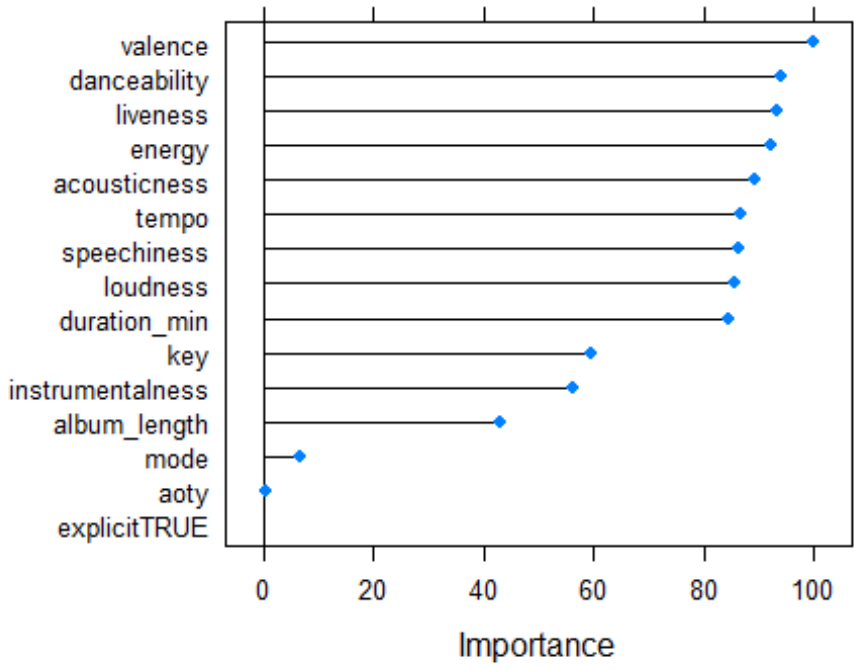
Here is classification with random forest

```
######################
# Random Forest
######################
rf1 <- train(as.factor(song_third_num) ~ ., data = songs_pred, method = "rf")
# Default in R
# Changes every time, average 35%
confusionMatrix(rf1) # table of percentages. Expect counts - x460
```

```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    1    2    3
##          1  8.2  9.8  8.5
##          2 11.2 10.2 10.7
##          3 11.9 12.1 17.4
##
##  Accuracy (average) : 0.3575
```

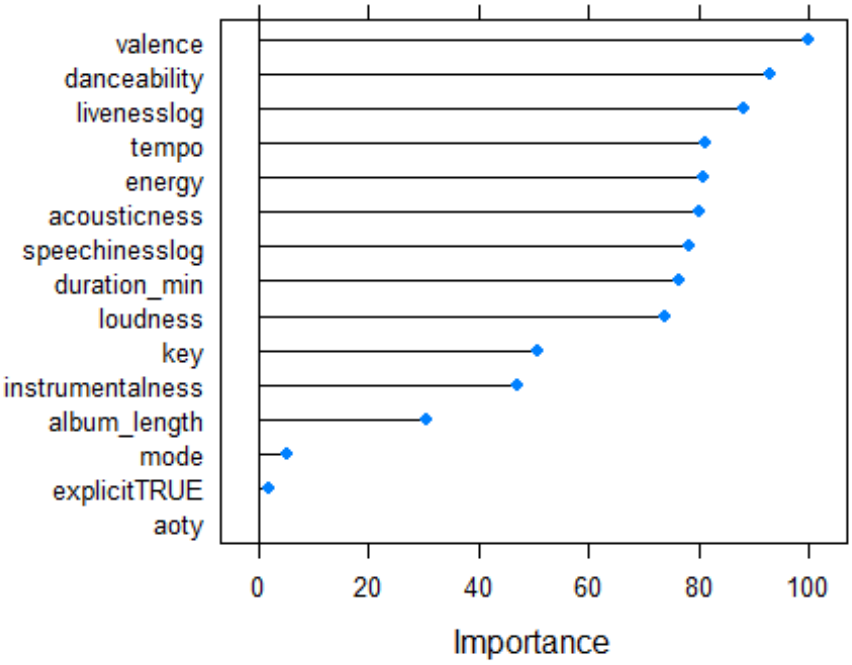
```
varImp(rf1)
```

```
## rf variable importance
##
##                   Overall
## valence          100.0000
## danceability      93.9704
## liveness          93.2434
## energy            92.0200
## acousticness      89.3201
## tempo             86.7210
## speechiness       86.1497
## loudness          85.6300
## duration_min      84.3809
## key               59.4759
## instrumentalness  56.1543
## album_length      42.8670
## mode               6.8608
## aoty               0.4457
## explicitTRUE       0.0000

plot(varImp(rf1))
```



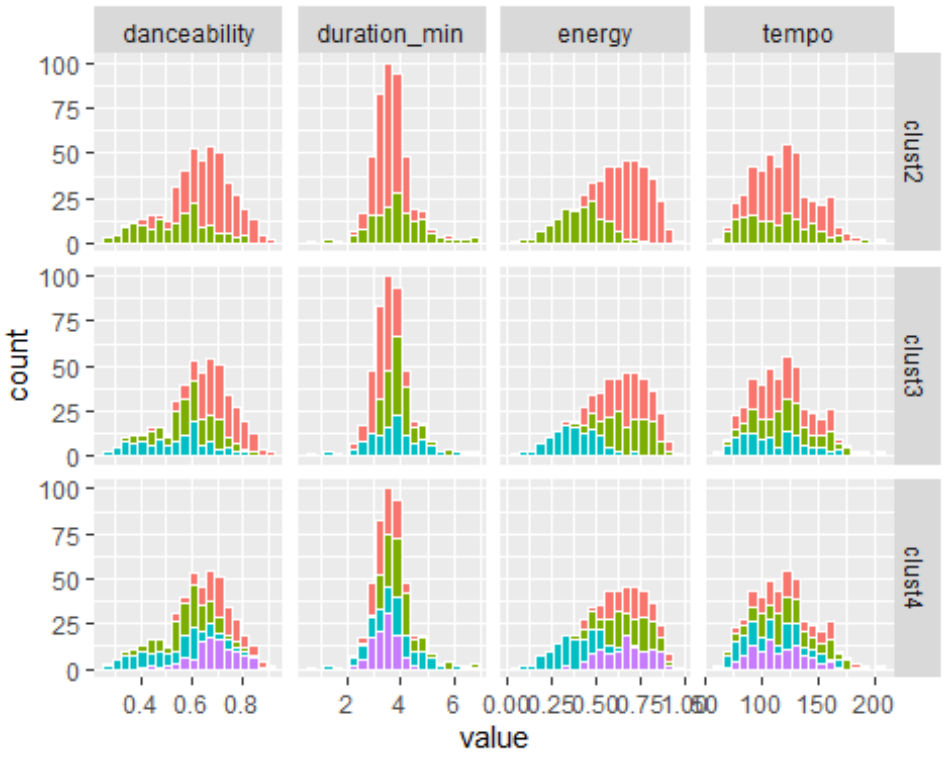

```
######################
# Random Forest with logged vars
######################
rf1log <- train(as.factor(song_third_num) ~ ., data = songs_predlog, method =
"rf")  # Default in R
```

```
# Always moving - 34.2%
confusionMatrix(rf1log) # table of percentages. Expect counts - x460

## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    1    2    3
##          1  9.6 11.2 10.3
##          2  9.0  9.7  8.4
##          3 12.0 13.1 16.7
##
##  Accuracy (average) : 0.3605

varImp(rf1log)

## rf variable importance
##
##                   Overall
## valence           100.000
## danceability       92.868
## livenesslog        88.117
## tempo              81.233
## energy             80.964
## acousticness       80.195
## speechinesslog     78.264
## duration_min       76.470
## loudness           74.026
## key                50.791
## instrumentalness   47.066
## album_length       30.738
## mode                5.328
## explicitTRUE        1.882
## aoty                0.000

plot(varImp(rf1log))
```

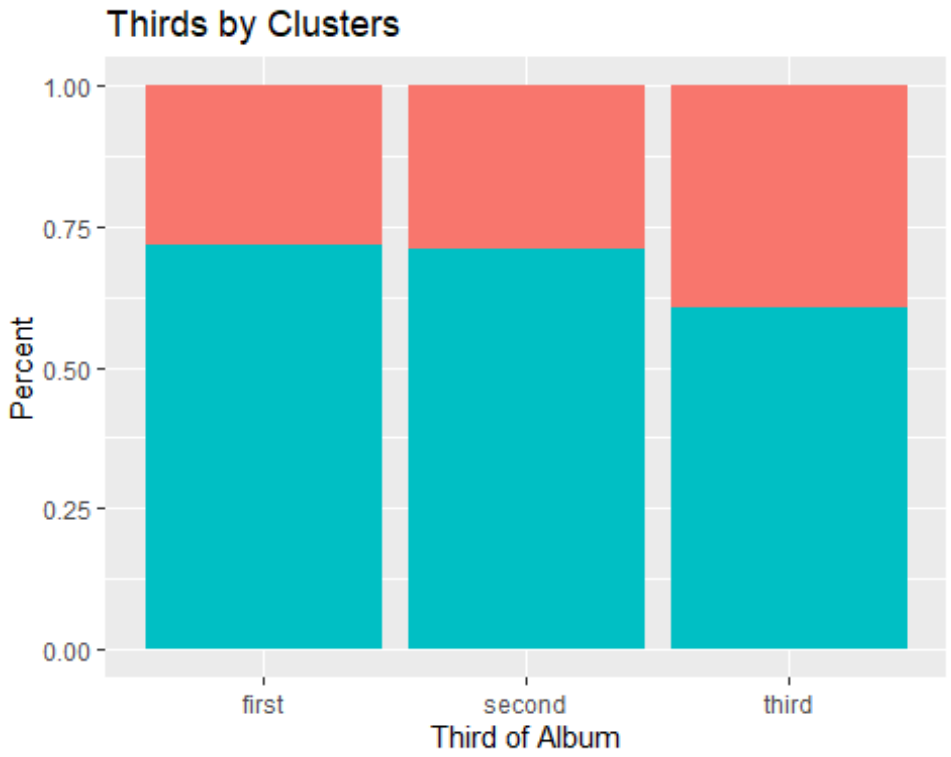


Unsupervised Learning - Clustering

```
# Clustering
songs_num %>%
  mutate(clust2 = kmeans(scale(songs_numlog),2)$cluster,
         clust3 = kmeans(scale(songs_numlog),3)$cluster,
         clust4 = kmeans(scale(songs_numlog),4)$cluster) %>%
  pivot_longer(c(danceability,duration_min,energy,tempo),"varname","value")
%>%

pivot_longer(c(clust2,clust3,clust4),names_to="numclust",values_to="clustnum"
) %>%
  mutate(numclust = factor(numclust, levels=c("clust2", "clust3","clust4")))
%>%
  ggplot() +
  geom_histogram(aes(value,fill=as.character(clustnum)),color="white", bins =
20, show.legend=F) +
  facet_grid(numclust~ varname, scales = "free")
```




```
# Visual
songs %>%
  mutate(clust = kmeans(scale(songs_numlog),2)$cluster) %>% # bring songs_num
clusters in
  ggplot()+geom_bar(aes(x=song_third, fill=as.factor(clust)),position="fill")
+
  theme(legend.position="none")+
  labs(x="Third of Album",y="Percent",title="Thirds by Clusters")
```

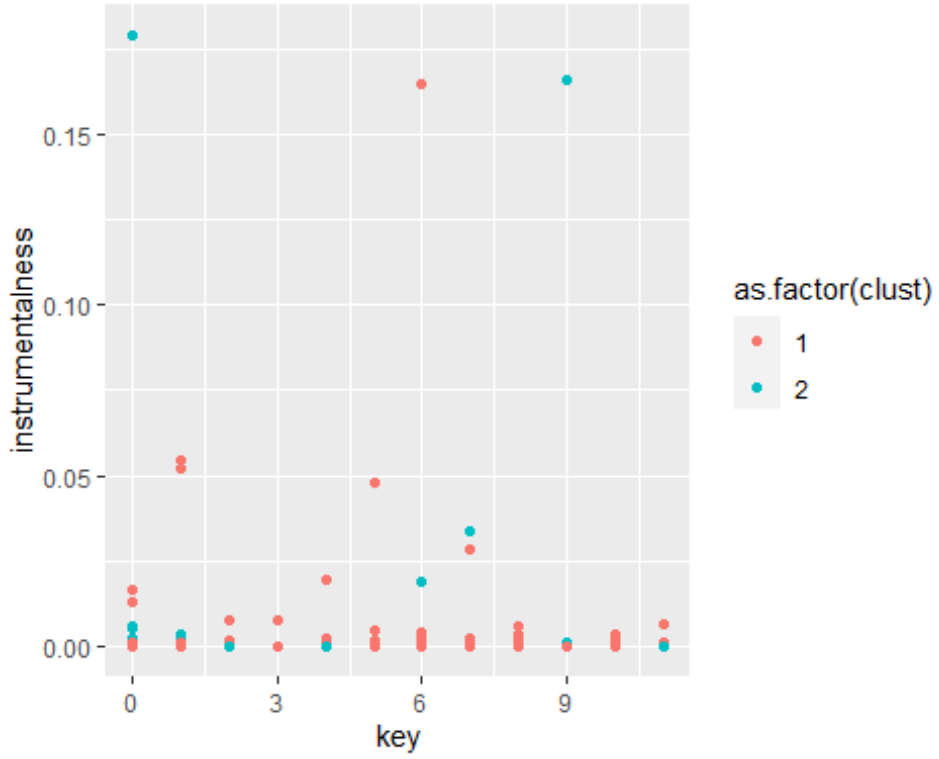



```
# what goes into clusters
set.seed(1)
songs %>%
  mutate(clust = kmeans(scale(songs_numlog),2)$cluster) %>% # centers for
group 1 and 2

pivot_longer(c(danceability,energy,key,loudness,mode,speechiness,acousticness
,instrumentalness,liveness,valence,tempo,explicit,duration_min,aoty,album_len
gth),names_to = "varname", values_to = "value") %>%
  group_by(clust, varname) %>%
  mutate(value_mean = mean(value)) %>%
  group_by(varname) %>%
  summarize(ssmod = sum((value-value_mean)^2),
            sstot = sum((value-mean(value))^2)) %>%
  mutate(r_sq = ssmod/sstot) %>%
  arrange(desc(r_sq))
```

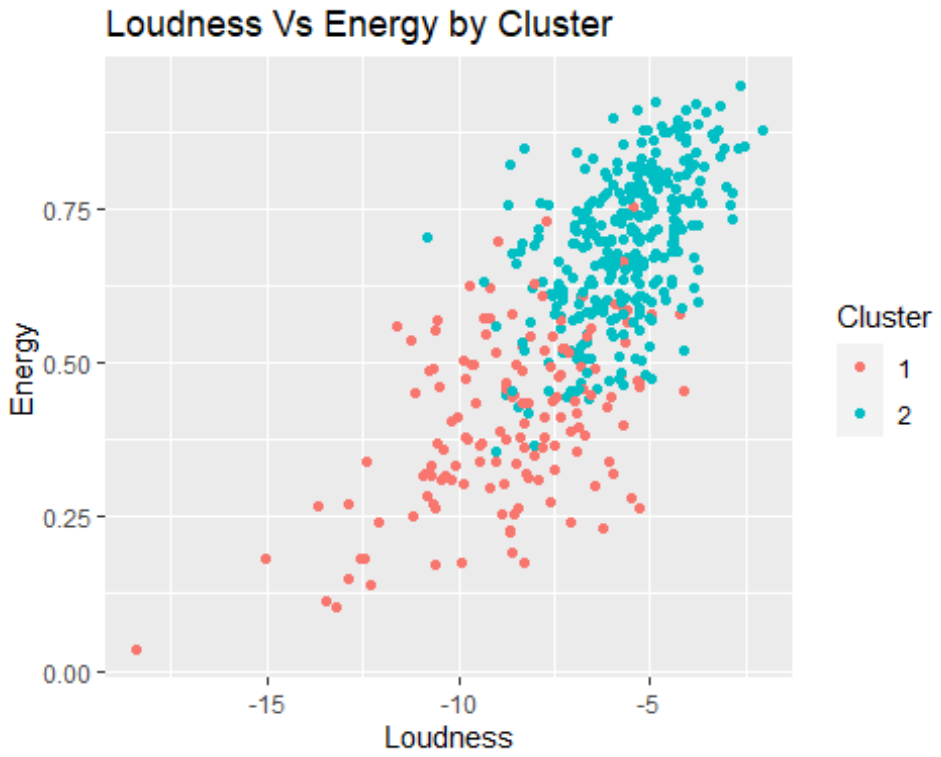
```
## # A tibble: 15 x 4
##    varname                ssmod       sstot  r_sq
##    <chr>                  <dbl>       <dbl> <dbl>
##  1 key                  5712.       5712.     1.00
##  2 explicit               49.1        49.2    0.998
##  3 instrumentalness        0.0964      0.0966 0.998
##  4 album_length         2440.       2469.     0.988
##  5 liveness                5.11        5.17   0.988
##  6 tempo              328615.     335564.     0.979
##  7 mode                   98.1       100.     0.977
```

```
##  8 speechiness           1.86        1.91   0.976
##  9 aoty                 62.7        64.8    0.968
## 10 duration_min        248.        257.     0.964
## 11 valence              15.6        20.1    0.776
## 12 danceability          5.90        7.68   0.768
## 13 loudness           1357.       2231.     0.608
## 14 energy                6.86       15.3    0.450
## 15 acousticness         18.5        42.9    0.431
```

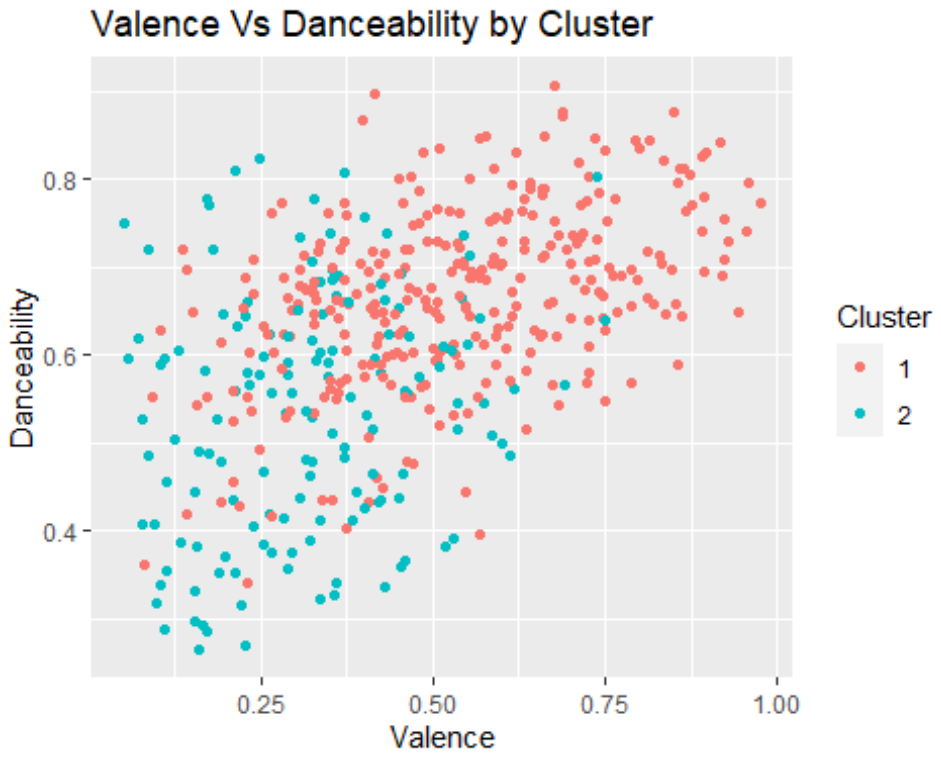
```
songs %>%
  mutate(clust = kmeans(scale(songs_numlog),2)$cluster) %>%
  ggplot() +
  geom_point(aes(x=key,y=instrumentalness,col=as.factor(clust)))
```



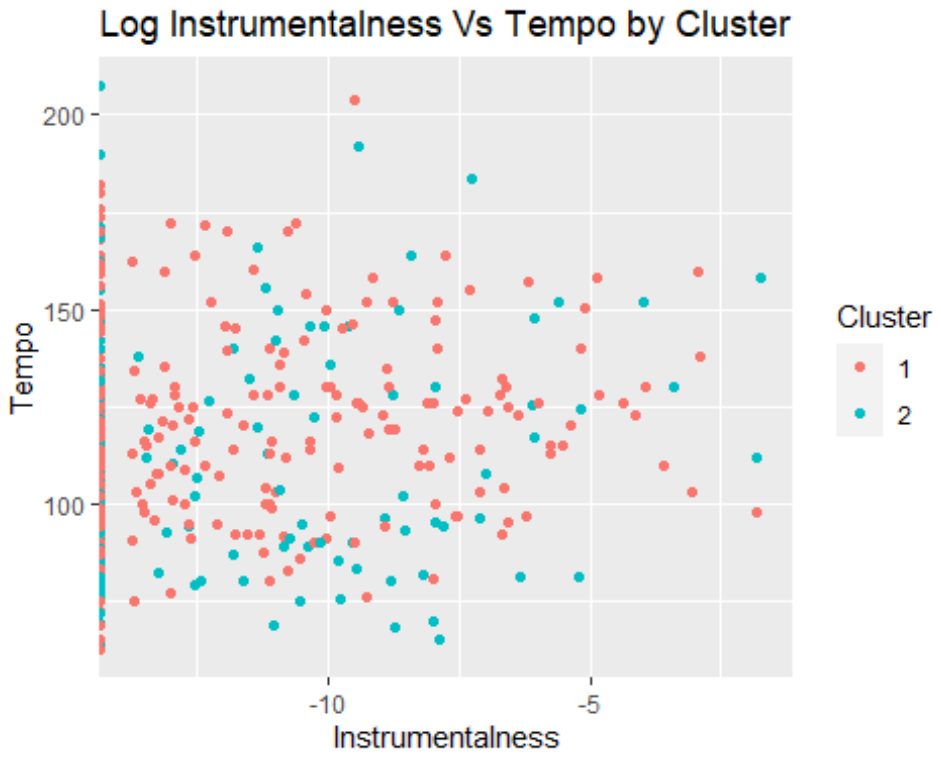

```
# This is the most revealing
songs %>%
  mutate(clust = kmeans(scale(songs_num),2)$cluster) %>%
  ggplot() +
  geom_point(aes(x=loudness,y=energy,col=as.factor(clust)))+
  labs(x="Loudness",y="Energy",title="Loudness Vs Energy by
Cluster",col="Cluster")
```




```
# They're very correlated
cor(songs$loudness,songs$energy)

## [1] 0.7234325

# High happy means high danceability
songs %>%
  mutate(clust = kmeans(scale(songs_numlog),2)$cluster) %>%
  ggplot() +
  geom_point(aes(x=valence,y=danceability,col=as.factor(clust)))+
  labs(x="Valence",y="Danceability",title="Valence Vs Danceability by
Cluster",col="Cluster")
```

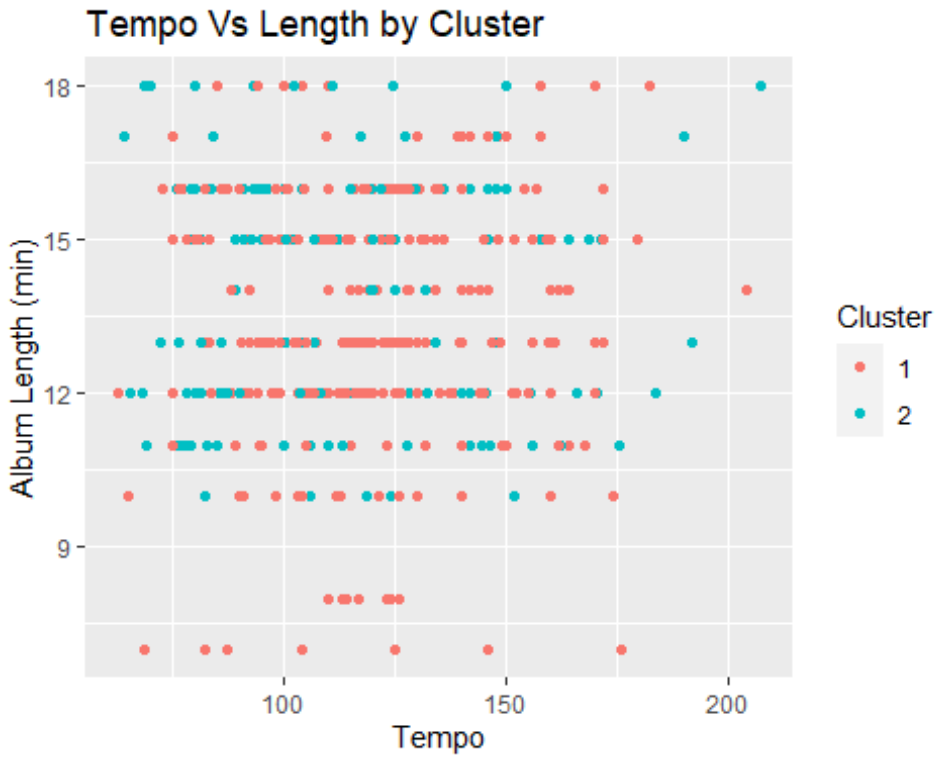



```
# Log looks better since instrumentalness is so low (just couldn't keep it
calculated)
songs %>%
  mutate(loginst = log(instrumentalness)) %>%
  mutate(clust = kmeans(scale(songs_numlog),2)$cluster) %>%
  ggplot() +
  geom_point(aes(x=loginst,y=tempo,col=as.factor(clust)))+
  labs(x="Instrumentalness",y="Tempo",title="Log Instrumentalness Vs Tempo by
Cluster",col="Cluster")
```




```
# Pretty evenly distributed on both parts
songs %>%
  mutate(clust = kmeans(scale(songs_numlog),2)$cluster) %>%
  ggplot() +
  geom_point(aes(x=track_placement,y=explicit,col=as.factor(clust)))+
  labs(x="Placement",y="Explicit",title="Placement vs Explicit by
Cluster",col="Cluster")
```

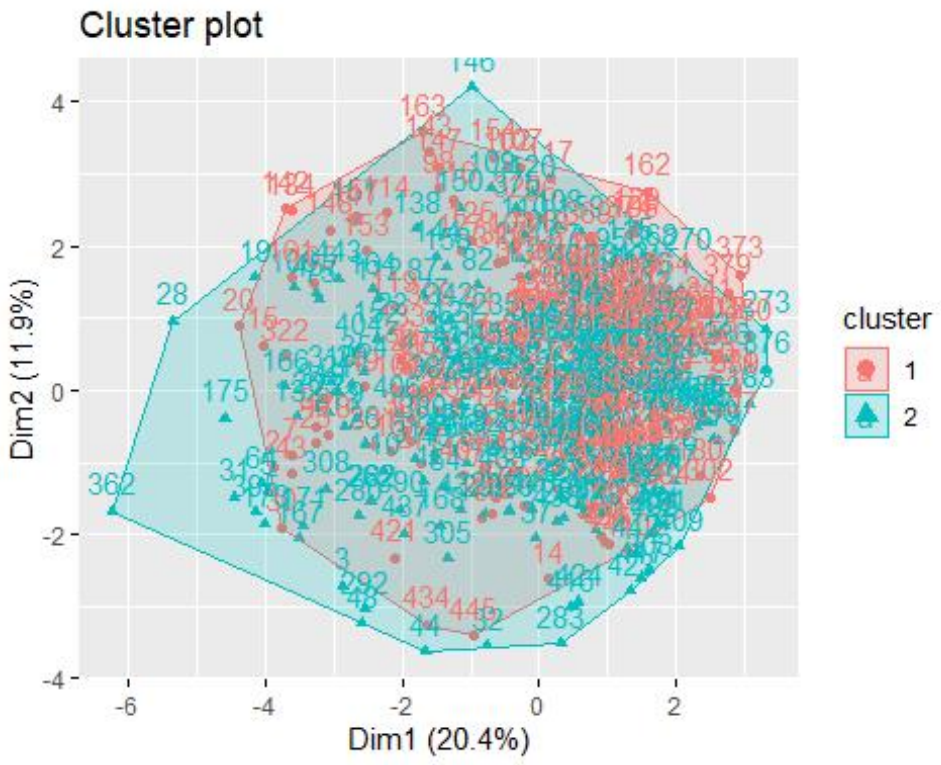

```
# Not helpful
songs %>%
  mutate(clust = kmeans(scale(songs_numlog),2)$cluster) %>%
  ggplot() +
  geom_point(aes(x=tempo,y=album_length,col=as.factor(clust))) +
  labs(x="Tempo",y="Album Length (min)",title="Tempo Vs Length by
Cluster",col="Cluster")
```



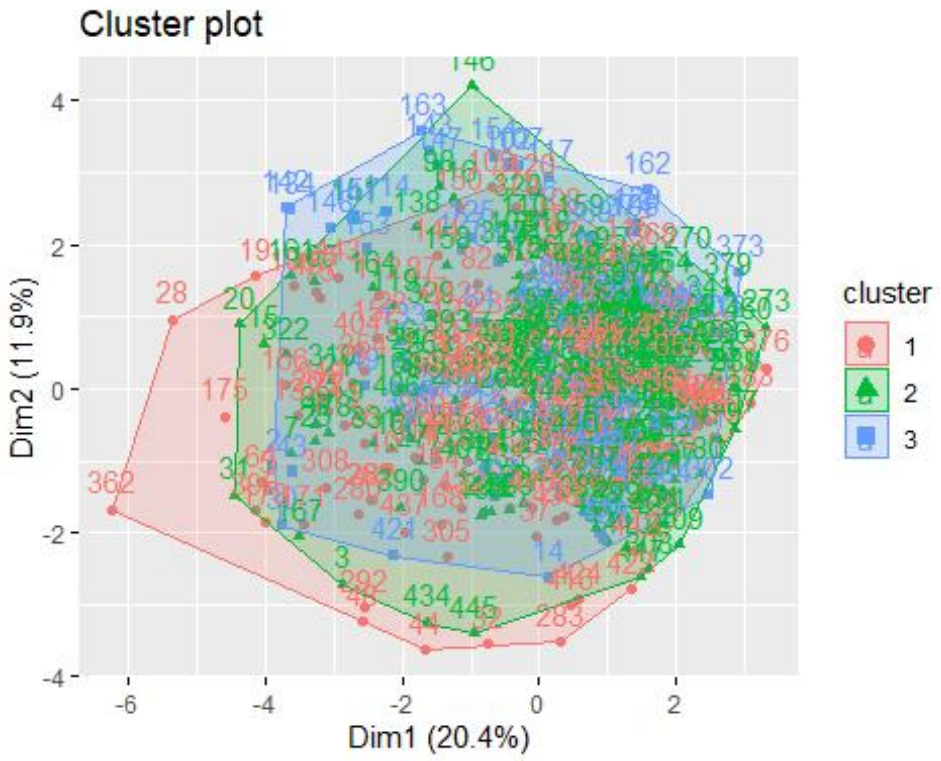

```
# I test for an association between song third and the two clusters. #
p=.0265 so there is sig ev of an association
chisq.test(table(songs2$third, kmeans(scale(songs_numlog),2)$cluster))

##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table(songs2$third, kmeans(scale(songs_numlog), 2)$cluster)
## X-squared = 4.1476, df = 1, p-value = 0.04169

table(songs2$third, kmeans(scale(songs_numlog),2)$cluster) %>% prop.table(1)

##
##             1         2
##   0 0.7118644 0.2881356
##   1 0.6060606 0.3939394

# but not sig evidence of a diff here because of smaller sample sizes
chisq.test(table(songs2$song_third, kmeans(scale(songs_numlog),2)$cluster))

##
##  Pearson's Chi-squared test
##
## data:  table(songs2$song_third, kmeans(scale(songs_numlog), 2)$cluster)
## X-squared = 5.4052, df = 2, p-value = 0.06703

table(songs2$song_third, kmeans(scale(songs_num),2)$cluster) %>%
prop.table(1)
```

```
##
##                   1         2
##   first  0.7152778 0.2847222
##   second 0.7218543 0.2781457
##   third  0.6121212 0.3878788
```

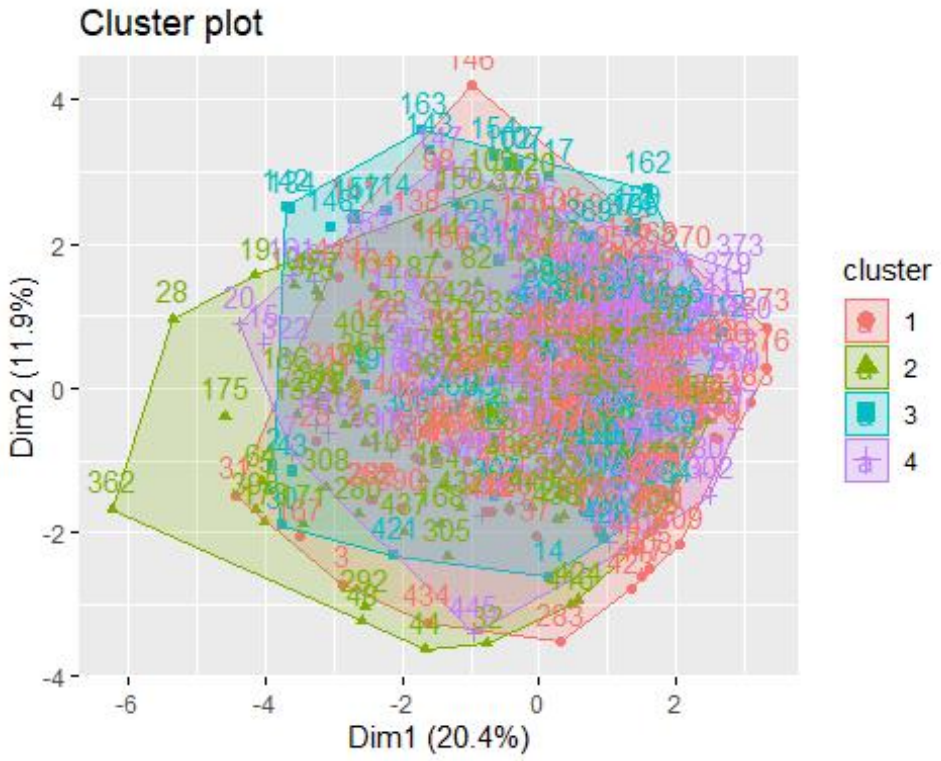
```
# fun cluster graph
songs_num %>% kmeans(2) %>% fviz_cluster(songs_numlog)
```




```
songs_num %>% kmeans(3) %>% fviz_cluster(songs_numlog) # maybe the best but not very good
```




```
songs_num %>% kmeans(4) %>% fviz_cluster(songs_numlog)
```



```
songs %>% mutate(cluster = (songs_numlog %>% scale() %>% kmeans(2))$cluster)
%>%
  dplyr::select(c(cluster, song_third_num)) %>% table() %>% prop.table(2)

##         song_third_num
## cluster         1         2         3
##       1 0.6944444 0.7019868 0.6000000
##       2 0.3055556 0.2980132 0.4000000
```

Logistic Regression

I performed logistic regression on whether or not an album would be given a Grammy Award for Album of the Year. I used albums for units of measurement instead of songs and logged the same two variables. Both logged and unlogged models produce the same final model - valence alone predicts AOTY where an increase in 1 unit of valence is associated with a 1.4078 decrease in the log odds of winning AOTY.

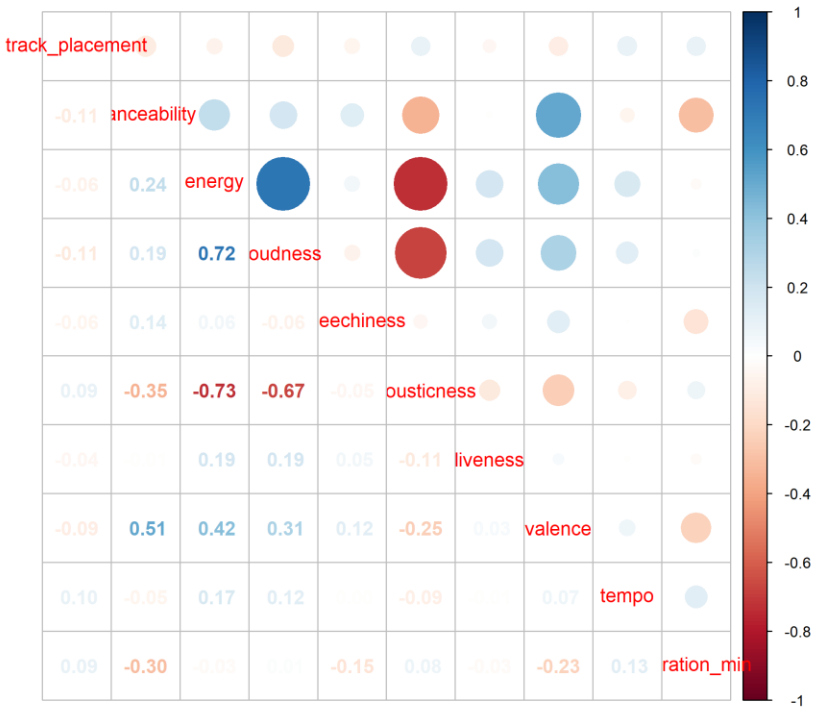
```
###################
# loop regression
###################
varlist <- names(grammy[,c(2:13,15)]) # get a list without id and group
pval_list <- matrix(NA,nrow=2,ncol=13) # every variable will have a p-val

for (i in seq_along(varlist)){
  # First, create null model needed to calculate p-val.
  # It needs to be done like this so it's comparable to the models made in
the loop
  base_blueprint <- as.formula(sprintf("aoty ~ 1"))
  base_model <- glm(formula = base_blueprint, family = binomial, data =
grammy)

  # code_blueprint lets me change the x variable in my glm model
  code_blueprint <- as.formula(sprintf("aoty ~ %s", varlist[i]))
  # create glmmodel with one variable each time
  glm_model <- glm(formula = code_blueprint, family = binomial, data =
grammy)

  # Find p-value by comparing each model to the null model. I want 2nd pval,
the one from hm_full.
  pval_list[1,i] <- varlist[i] # first row is col name
  pval_list[2,i] <- anova(base_model, glm_model, test =
'Chisq')$"Pr(>Chi)"[2]
}
min(pval_list)
# valence has smallest p-val

## add valence ##
varlist <- names(grammy[,c(2:9,11,12,13,15)])
pval_list <- matrix(NA,nrow=2,ncol=12)
```

```r
for (i in seq_along(varlist)){
  base_blueprint <- as.formula(sprintf("aoty ~ valence"))
  base_model <- glm(formula = base_blueprint, family = binomial, data =
grammy)
  code_blueprint <- as.formula(sprintf("aoty ~ valence + %s", varlist[i]))
  glm_model <- glm(formula = code_blueprint, family = binomial, data =
grammy)
  pval_list[1,i] <- varlist[i]
  pval_list[2,i] <- anova(base_model, glm_model, test =
'Chisq')$"Pr(>Chi)"[2]
}
min(pval_list)

model <- glm(aoty ~ valence, data=grammy)
summary(model) # aic = 31.646



##############################################

# Do it again with grammylog data


##################
# loop regression
##################
varlistlog <- names(grammylog[,c(2:13,15)]) # get a list without id and group
pval_list <- matrix(NA,nrow=2,ncol=13) # every variable will have a p-val

for (i in seq_along(varlistlog)){
  base_blueprint <- as.formula(sprintf("aoty ~ 1"))
  base_model <- glm(formula = base_blueprint, family = binomial, data =
grammylog)
  code_blueprint <- as.formula(sprintf("aoty ~ %s", varlistlog[i]))
  glm_model <- glm(formula = code_blueprint, family = binomial, data =
grammylog)
  pval_list[1,i] <- varlistlog[i]
  pval_list[2,i] <- anova(base_model, glm_model, test =
'Chisq')$"Pr(>Chi)"[2]
}
min(pval_list)
# valence has smallest p-val

## add valence ##
varlistlog <- names(grammylog[,c(2:9,11,12,13,15)])
pval_list <- matrix(NA,nrow=2,ncol=12)

for (i in seq_along(varlistlog)){
  base_blueprint <- as.formula(sprintf("aoty ~ valence"))
```

```
  base_model <- glm(formula = base_blueprint, family = binomial, data =
grammylog)
  code_blueprint <- as.formula(sprintf("aoty ~ valence + %s", varlistlog[i]))
  glm_model <- glm(formula = code_blueprint, family = binomial, data =
grammylog)
  pval_list[1,i] <- varlistlog[i]
  pval_list[2,i] <- anova(base_model, glm_model, test =
'Chisq')$"Pr(>Chi)"[2]
}
min(pval_list) # stop here

modellog <- glm(aoty ~ valence, data=grammylog)
#summary(modellog) # aic = 31.646
```

The final model

```
modellog <- glm(aoty ~ valence, data=grammylog)

summary(modellog)

Call:

glm(formula = aoty ~ valence, data = grammylog)



Deviance Residuals:

     Min        1Q    Median        3Q       Max
-0.41809  -0.22803 -0.13033  0.05648  0.81914

Coefficients:

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.8311     0.2884   2.882   0.0069 **
valence      -1.4078     0.6016  -2.340   0.0255 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.1292112)

    Null deviance: 4.9714  on 34  degrees of freedom
Residual deviance: 4.2640  on 33  degrees of freedom
AIC: 31.646

Number of Fisher Scoring iterations: 2

# Correlation plot
```



#Correlation plot with logged variables