Linear Regression Analysis of Trending TikTok Tracks

Fatemeh Shayesteh

1/15/2022

I. Introduction

TikTok, known in China as Douyin, is a video-focused social networking service owned by Chinese company ByteDance. It hosts a variety of short-form user videos, from genres like pranks, stunts, tricks, jokes, dance, and entertainment with durations from 15 seconds to three minutes. TikTok has dethroned Google as the most popular domain in 2021, according to Cloudflare's 2021 Year in Review for internet traffic.

In this project, I used a public date set shared by Kaggle. The dataset contains trending tracks featured on TikTok. This dataset was created by Team Dan and contains around 7000 records along with liveness score, technical information and other features such as, duration, danceability, energy, and loudness. This dataset includes 22 variables. You can see the detail description of these variables in Appendix. Looking at few records of the dataset gives some first impressions of the data like features and data types of the features.

```
head(TikTok)

##                  track_id                             track_name
## 1 3BZEcbdtXQSo7OrvKRJ6mb         MONTERO (Call Me By Your Name)
## 2 4iJyoBOLtHqaGxP12qzhQI Peaches (feat. Daniel Caesar & Giveon)
## 3 3Ofmpyhv5UAQ70mENzB277                 Astronaut In The Ocean
## 4 4J92EUjxBf8cj3yMr3M6B5                           In The Night
## 5 748mdHapucXQri7IAO8yFK               Kiss Me More (feat. SZA)
## 6 7MAibcTli4IisCtbHKrGMh                    Leave The Door Open
##                 artist_id   artist_name               album_id duration
## 1 7jVv8c5Fj3E9VhNjxT4snq     Lil Nas X 5iZytG7j5DDp9RlsmkGI97   137875
## 2 1uNFoZAHBGtllmzznpCI3s Justin Bieber 5dGWwsZ9iB2Xc3UKR0gif2   198081
## 3 1uU7g3DNSbsu0QjSEqZtEd   Masked Wolf 7vus4Q8r5DS2Dl1JClxEsA   132780
## 4 2tndYCXQneCV4jtoWRwVpz Lucas Estrada 1y8WgXMkztAiCkcGJh4ENO   162616
## 5 5cj0lLjcoR7YOSnhnX0Po5      Doja Cat 1OnzqJTL9bwe4kvaLxRYxt   208866
## 6 0du5cEVh5yTK9QJze8zA0C    Bruno Mars 7dfPqXck6BB9wpThrVYBss   242096
##   release_date popularity danceability energy key loudness mode speechines
s
## 1   2021-03-26         79        0.610  0.508   8   -6.682    0      0.152
0
## 2   2021-03-19         99        0.677  0.696   0   -6.181    1      0.119
0
## 3   2021-01-06         96        0.778  0.695   4   -6.865    0      0.091
3
## 4   2021-06-04          0        0.726  0.631   5   -7.083    0      0.062
5
```

```
## 5   2021-04-09         98        0.762  0.701   8   -3.541    1      0.028
6
## 6   2021-03-05         96        0.586  0.616   5   -7.964    1      0.032
4
##   acousticness instrumentalness liveness valence   tempo           playli
st_id
## 1        0.297         0.00e+00   0.3840   0.758 178.818 65LdqYCLcsV0lJoxp
eQ6fW
## 2        0.321         0.00e+00   0.4200   0.464  90.030 65LdqYCLcsV0lJoxp
eQ6fW
## 3        0.175         0.00e+00   0.1500   0.472 149.996 65LdqYCLcsV0lJoxp
eQ6fW
## 4        0.154         3.96e-05   0.2240   0.629 123.996 65LdqYCLcsV0lJoxp
eQ6fW
## 5        0.235         1.58e-04   0.1230   0.742 110.968 65LdqYCLcsV0lJoxp
eQ6fW
## 6        0.182         0.00e+00   0.0927   0.719 148.088 65LdqYCLcsV0lJoxp
eQ6fW
##                                    playlist_name
## 1 TikTok Songs 2021 Tik Tok Hits âš¡Summer 2021
## 2 TikTok Songs 2021 Tik Tok Hits âš¡Summer 2021
## 3 TikTok Songs 2021 Tik Tok Hits âš¡Summer 2021
## 4 TikTok Songs 2021 Tik Tok Hits âš¡Summer 2021
## 5 TikTok Songs 2021 Tik Tok Hits âš¡Summer 2021
## 6 TikTok Songs 2021 Tik Tok Hits âš¡Summer 2021
```

B. Aims.

The attraction to TikTok is that anyone can create whatever they wish in a matter of few minutes and attract millions of views. In order to gain a deep understanding of users and reflect their interest in product design and product discussions, in this project, I aim to identify the predictors that predict the popularity of TikTok tracks. In this project, I aim to study the popularity of TikTok artistic tracks and address this research question: RQ1: what variables do predict the popularity of TikTok trending tracks? H1a: TikTok trending tracks in dancing genre are more popular than tracks in other genres. H1b: TikTok trending tracks with higher tempo are more popular than tracks with lower tempo.

```
md1 = lm(popularity~valence+danceability+ energy + instrumentalness + loudnes
s + tempo)
```

IV. Discussion

There appears to be a positive relationship between instrumentallness,loudness, danceabllity, and the popularity of TikTok tracks. The limitations of this analysis include that the data include all popular tracks regardless of the fact that if the tracks have been originally recorded for the TikTok posting, or not. Some popular tracks have been recorded in professional setting and comparing them with tracks that are popular but have been specifically recorded for posting on TikTok is problematic. There are many variables that could account for popularity other than the variables that are available in our data.

My analysis showed that there are a considerable number of influential cases that affect the regression results. After we removed those influential cases, the R2 increased from 0.03 to 075. In other words, the greater amount of variability in tracks' popularity was explained by our predictors after we removed the influential cases.

Using different methods, I decided to remove the variables energy and valence from our model because the second model was the better fit based on different criteria. I also did some analysis on adding the interaction of variables. The findings indicated that the interaction of instrumentalness and loudness could help explaining the variability in popularity. Looking at the analysis of variance table indicated that we reject the null hypotheses and retain the interaction in our model because the p-value is <0.05 (0.04073).

model <- lm(popularity~danceability + loudness + instrumentalness + tempo

Although, the recent model very well explained the variation in popularity, as in our data exploratory, the algorithm suggested that most our explanatory variables have non-linear association with our response variable. It seems that a non-linear relation could predict our dependent variable more accurately. As such, I fitted a model with the squared root of loudness and tempo, two variables that the algorithm suggested to have a nonlinear relation with popularity. As the p-value was not small, I fail to reject the null hypothesis and had to drop the squared root of tempo. I also filed to reject the null hypothesis to add the interaction of instrumentallness and loudness. The only issue was that although this model was signifcant and the p value was significantly small, the R-squared decreased.

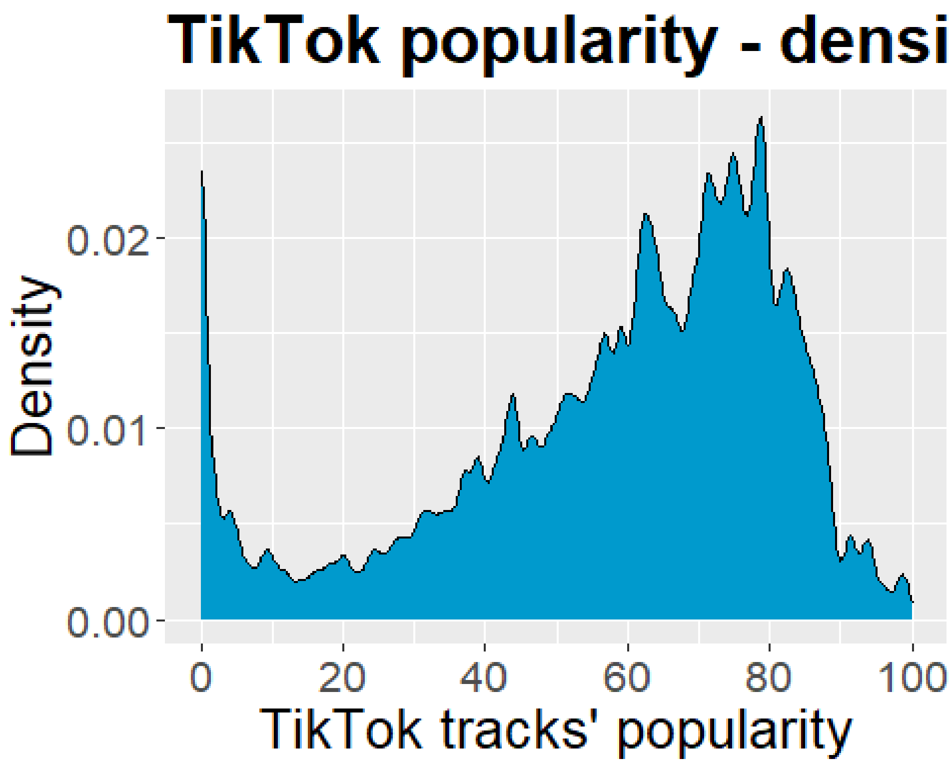
model1 <- lm(popularity~danceability + loudness.c + instrumentalness + tempo.c+ I(loudness.c^2))

V. Appendix

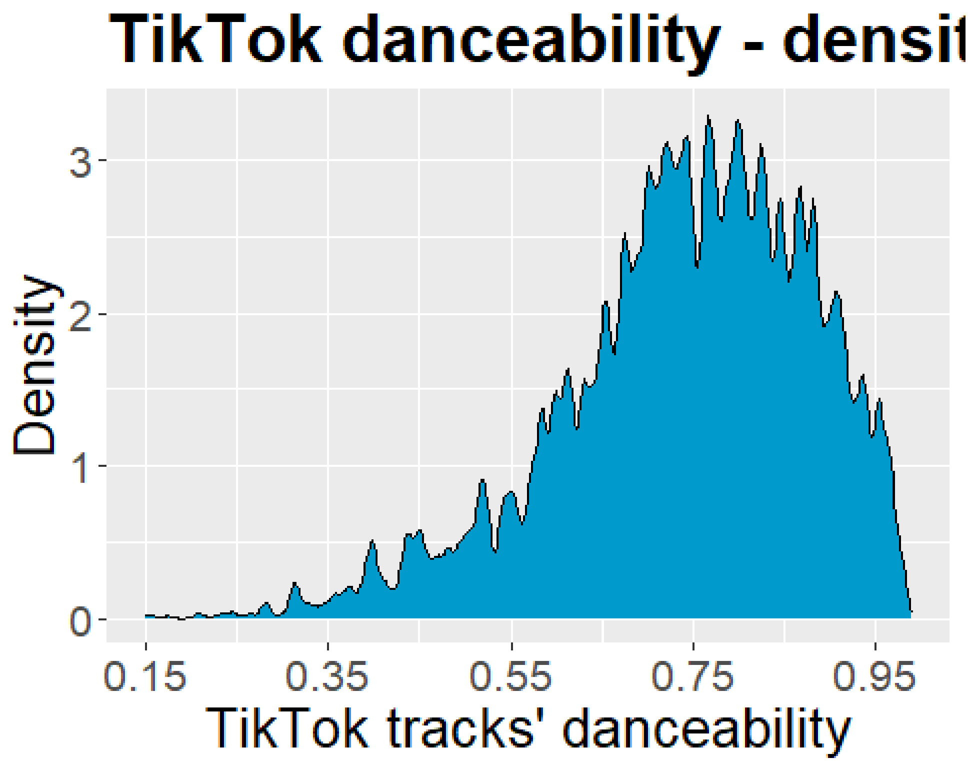
Data Exploratory

###Looking at distributions of numeric features First, I explore the feature distribution of variables using density plots.

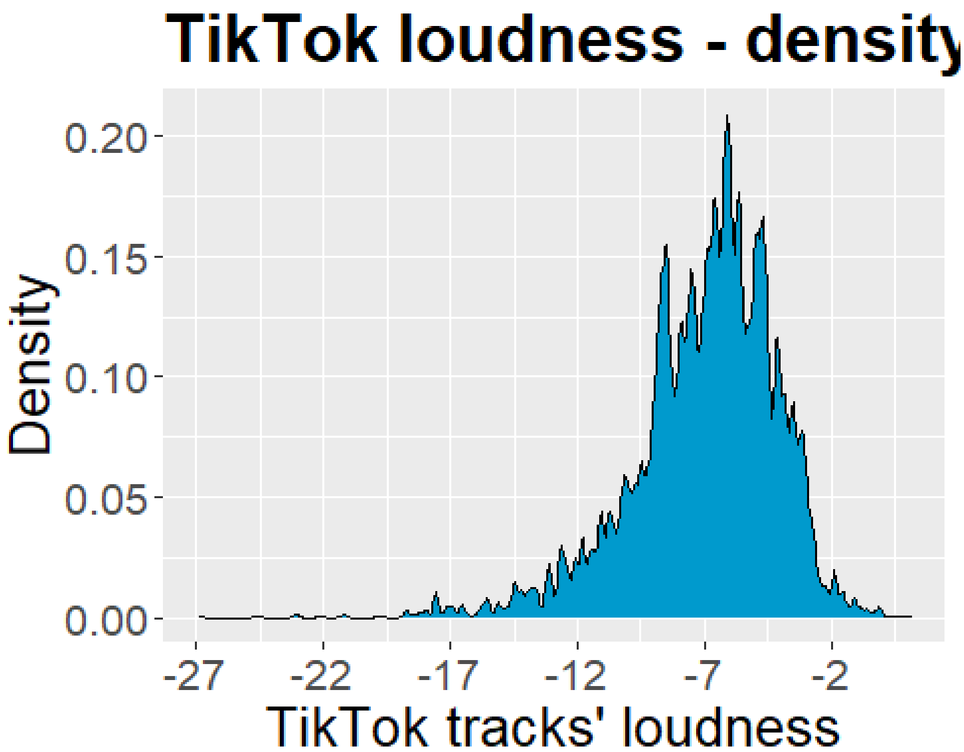
```
TikTok.data %>%
  ggplot(aes(x = popularity)) +
  geom_density(adjust = 1/5, color = "black", fill = "deepskyblue3") +
  xlab("TikTok tracks' popularity") +
  ylab("Density") +
  scale_x_continuous(breaks = seq(0,100,20)) +
  ggtitle("TikTok popularity - density plot") +
  theme(axis.text = element_text(size = 16),
        axis.title = element_text(size = 20),
        plot.title = element_text(size = 25, face = "bold"))
```



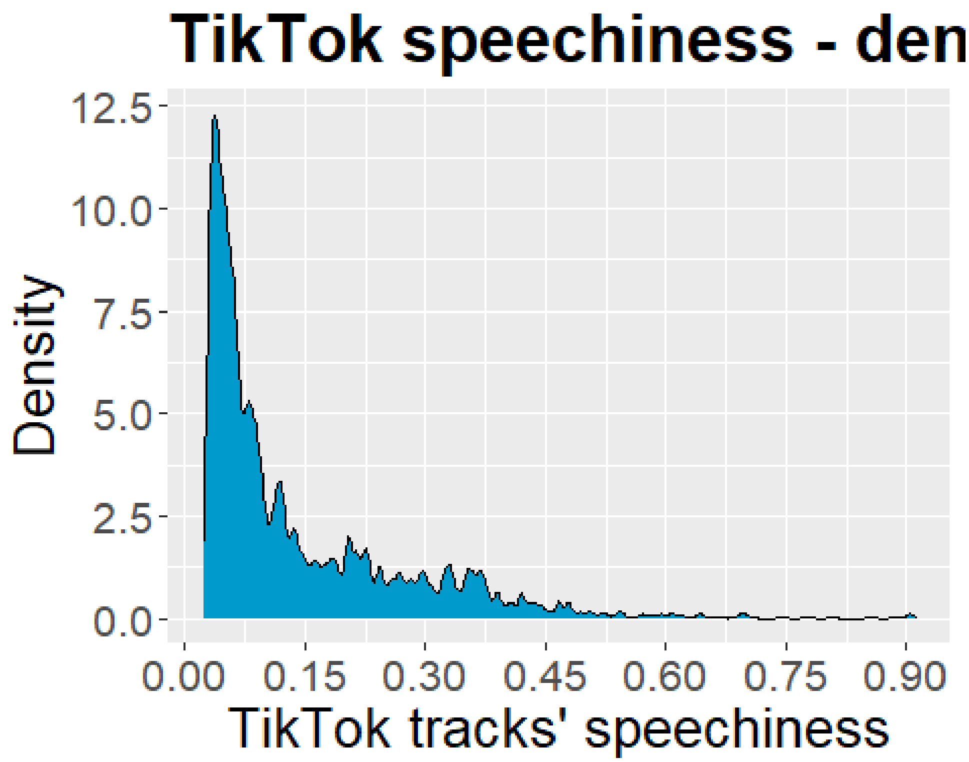

```
###The plot shows that only a small number of tracks have the popularity scor
e greater than 90.

TikTok.data %>%
  ggplot(aes(x = danceability)) +
  geom_density(adjust = 1/5, color = "black", fill = "deepskyblue3") +
  xlab("TikTok tracks' danceability") +
  ylab("Density") +
  scale_x_continuous(breaks = seq(0.15,1,0.2)) +
  ggtitle("TikTok danceability - density plot") +
  theme(axis.text = element_text(size = 16),
        axis.title = element_text(size = 20),
        plot.title = element_text(size = 25, face = "bold"))
```



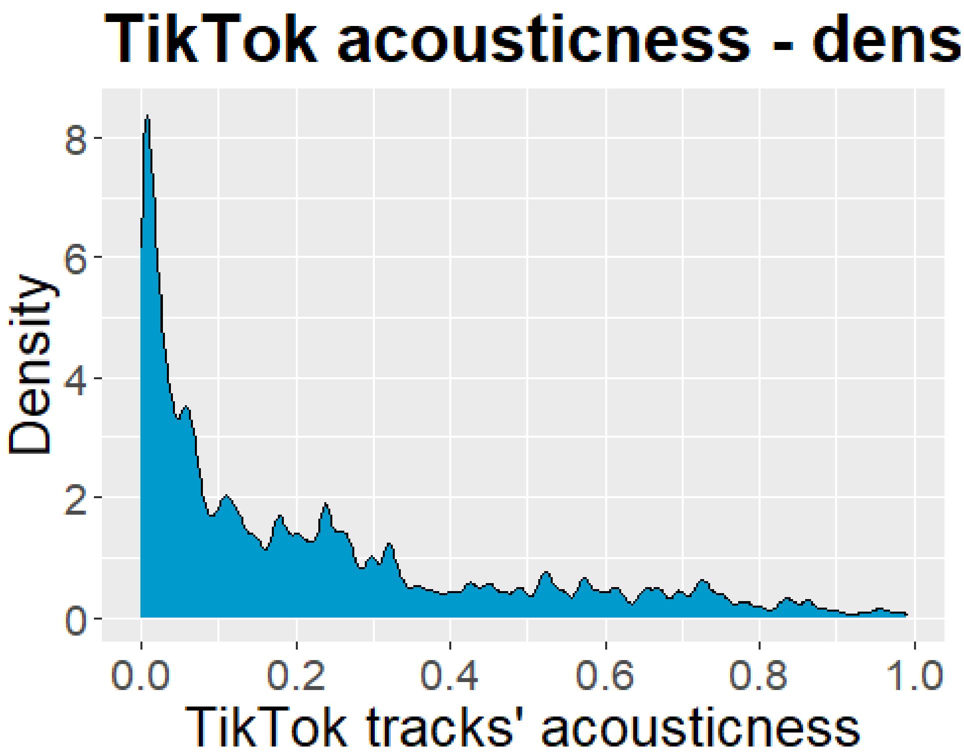

```
### The plot shows that only the small number of tracks are not  suitable for dancing.

TikTok.data %>%
  ggplot(aes(x = loudness)) +
  geom_density(adjust = 1/5, color = "black", fill = "deepskyblue3") +
  xlab("TikTok tracks' loudness") +
  ylab("Density") +
  scale_x_continuous(breaks = seq(-27,1.5,5)) +
  ggtitle("TikTok loudness - density plot") +
  theme(axis.text = element_text(size = 16),
        axis.title = element_text(size = 20),
        plot.title = element_text(size = 25, face = "bold"))
```



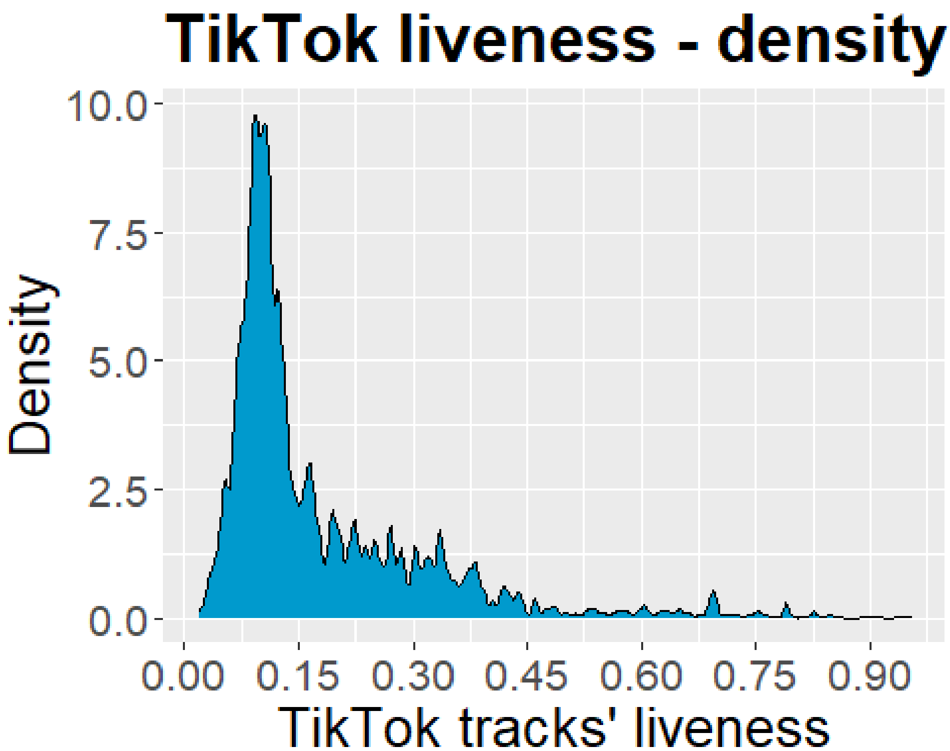

```
###The plot shows that the majority of tracks' loudness is between -10 and -2

TikTok.data %>%
  ggplot(aes(x = speechiness)) +
  geom_density(adjust = 1/5, color = "black", fill = "deepskyblue3") +
  xlab("TikTok tracks' speechiness") +
  ylab("Density") +
  scale_x_continuous(breaks = seq(0,1,0.15)) +
  ggtitle("TikTok speechiness - density plot") +
  theme(axis.text = element_text(size = 16),
        axis.title = element_text(size = 20),
        plot.title = element_text(size = 25, face = "bold"))
```



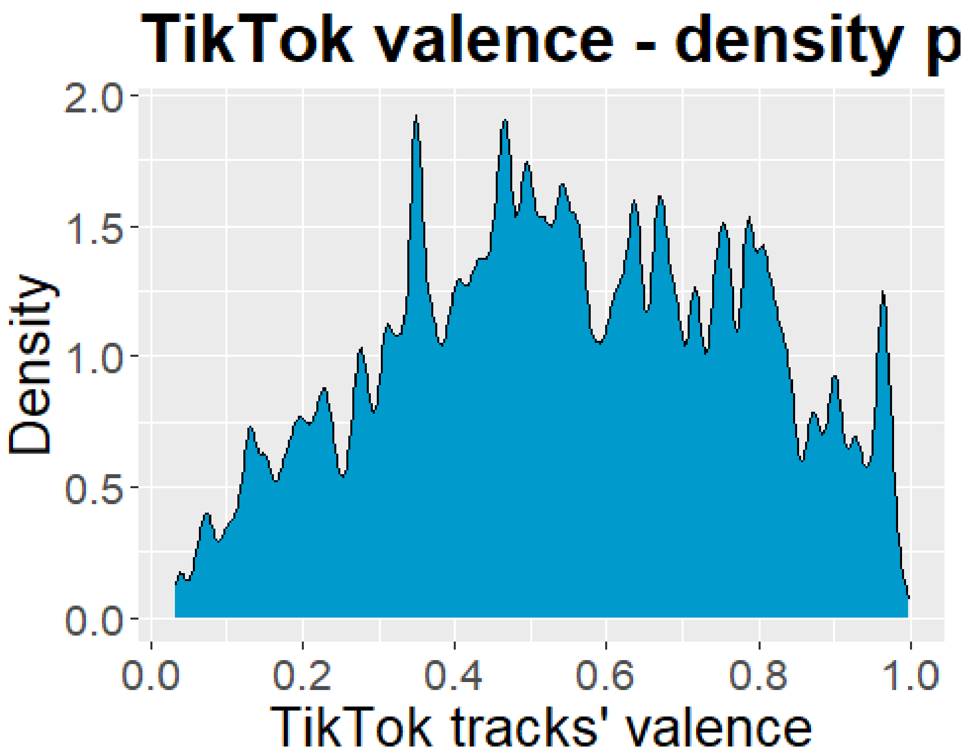

```
###The plot shows that the majority of tracks don't have the spoken words.

TikTok.data %>%
  ggplot(aes(x = acousticness)) +
  geom_density(adjust = 1/5, color = "black", fill = "deepskyblue3") +
  xlab("TikTok tracks' acousticness") +
  ylab("Density") +
  scale_x_continuous(breaks = seq(0,1,0.2)) +
  ggtitle("TikTok acousticness - density plot") +
  theme(axis.text = element_text(size = 16),
        axis.title = element_text(size = 20),
        plot.title = element_text(size = 25, face = "bold"))
```



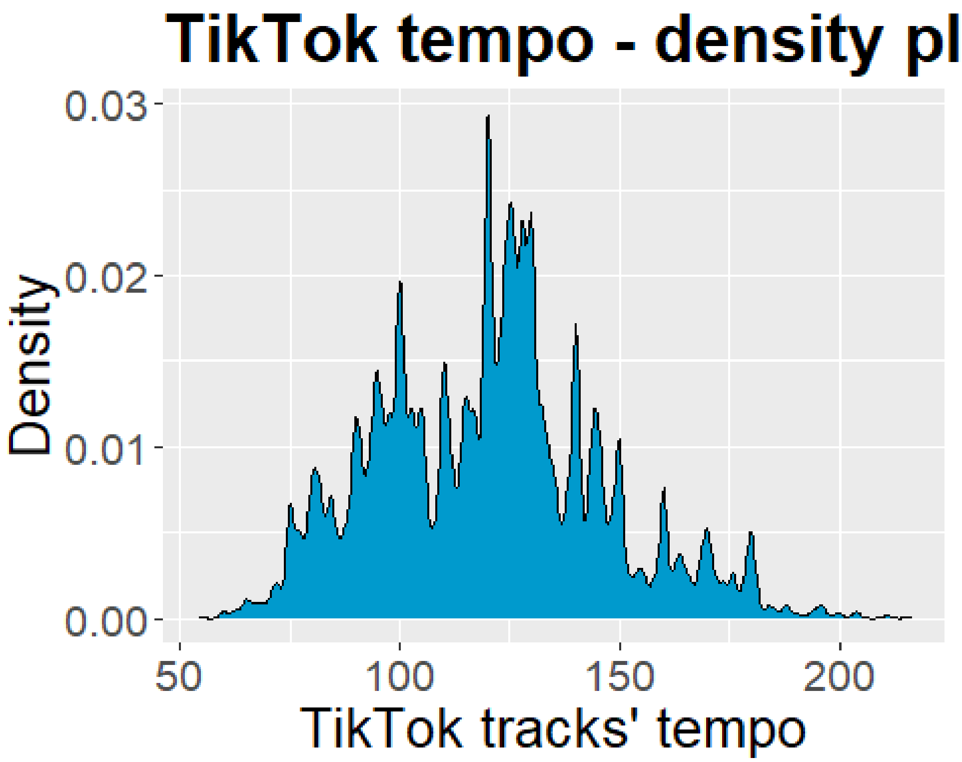

```
###It shows the small number of tracks are acoustic.

TikTok.data %>%
  ggplot(aes(x = liveness)) +
  geom_density(adjust = 1/5, color = "black", fill = "deepskyblue3") +
  xlab("TikTok tracks' liveness") +
  ylab("Density") +
  scale_x_continuous(breaks = seq(0,1,0.15)) +
  ggtitle("TikTok liveness - density plot") +
  theme(axis.text = element_text(size = 16),
        axis.title = element_text(size = 20),
        plot.title = element_text(size = 25, face = "bold"))
```



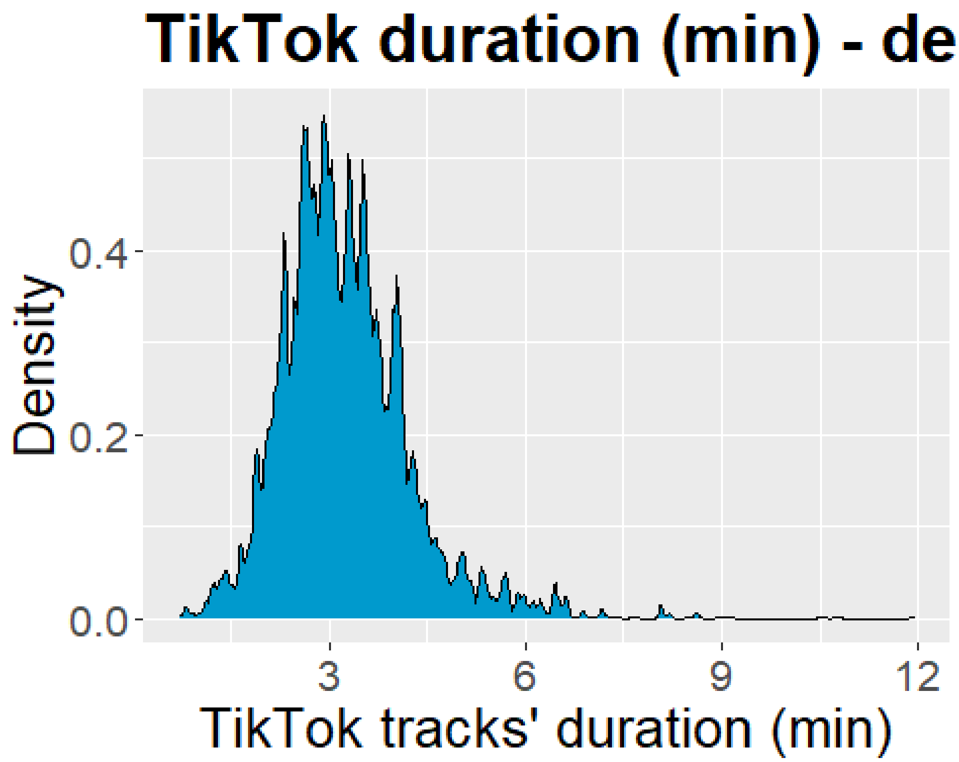

```
##The plot shows that most tracks lack the presence of an audience in the rec
ording.

TikTok.data %>%
  ggplot(aes(x = valence)) +
  geom_density(adjust = 1/5, color = "black", fill = "deepskyblue3") +
  xlab("TikTok tracks' valence") +
  ylab("Density") +
  scale_x_continuous(breaks = seq(0,1,0.2)) +
  ggtitle("TikTok valence - density plot") +
  theme(axis.text = element_text(size = 16),
        axis.title = element_text(size = 20),
        plot.title = element_text(size = 25, face = "bold"))
```




```
###The plot shows that only a small number of tracks have the valence score l
ess than 0.2.

TikTok.data %>%
  ggplot(aes(x = tempo)) +
  geom_density(adjust = 1/5, color = "black", fill = "deepskyblue3") +
  xlab("TikTok tracks' tempo") +
  ylab("Density") +
  scale_x_continuous(breaks = seq(0,250,50)) +
  ggtitle("TikTok tempo - density plot") +
  theme(axis.text = element_text(size = 16),
        axis.title = element_text(size = 20),
        plot.title = element_text(size = 25, face = "bold"))
```




```
##The plot shows that the majority of popular tracks have the tempo score bet
ween 100-150.

TikTok.data %>%
  ggplot(aes(x = duration_mins)) +
  geom_density(adjust = 1/5, color = "black", fill = "deepskyblue3") +
  xlab("TikTok tracks' duration (min)") +
  ylab("Density") +
  scale_x_continuous(breaks = seq(0,15,3)) +
  ggtitle("TikTok duration (min) - density plot") +
  theme(axis.text = element_text(size = 16),
        axis.title = element_text(size = 20),
        plot.title = element_text(size = 25, face = "bold"))
```

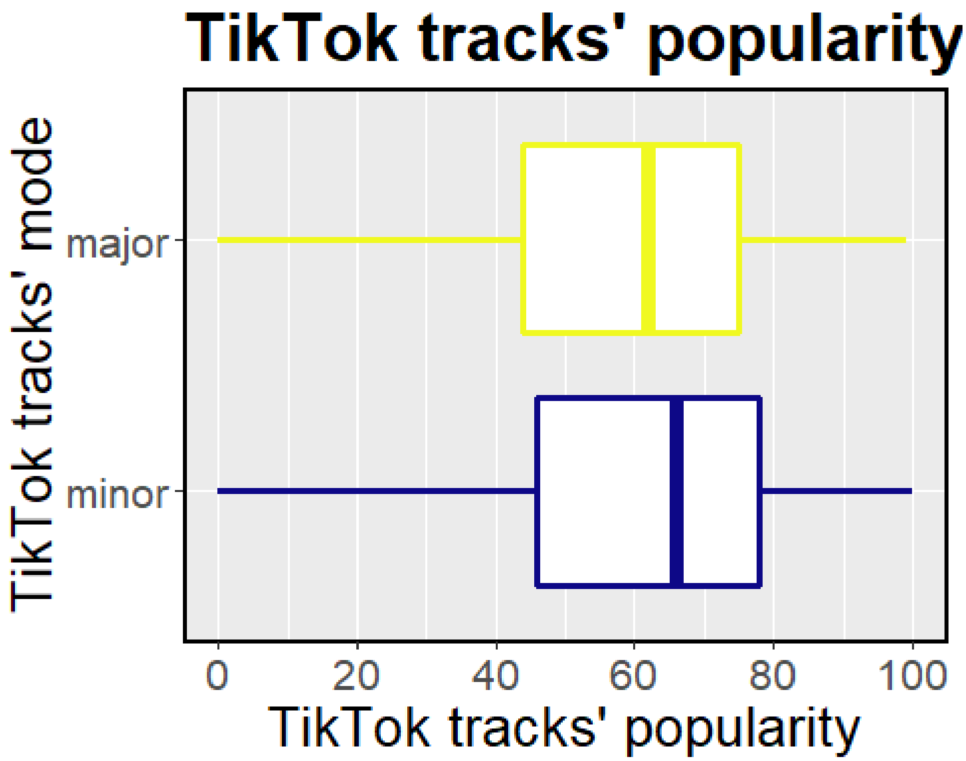



```
###The plot shows that the majority of tracks are about 3 minute long. Only a afew tracks are longer than 7 minutes.
##################################################################
##############
```

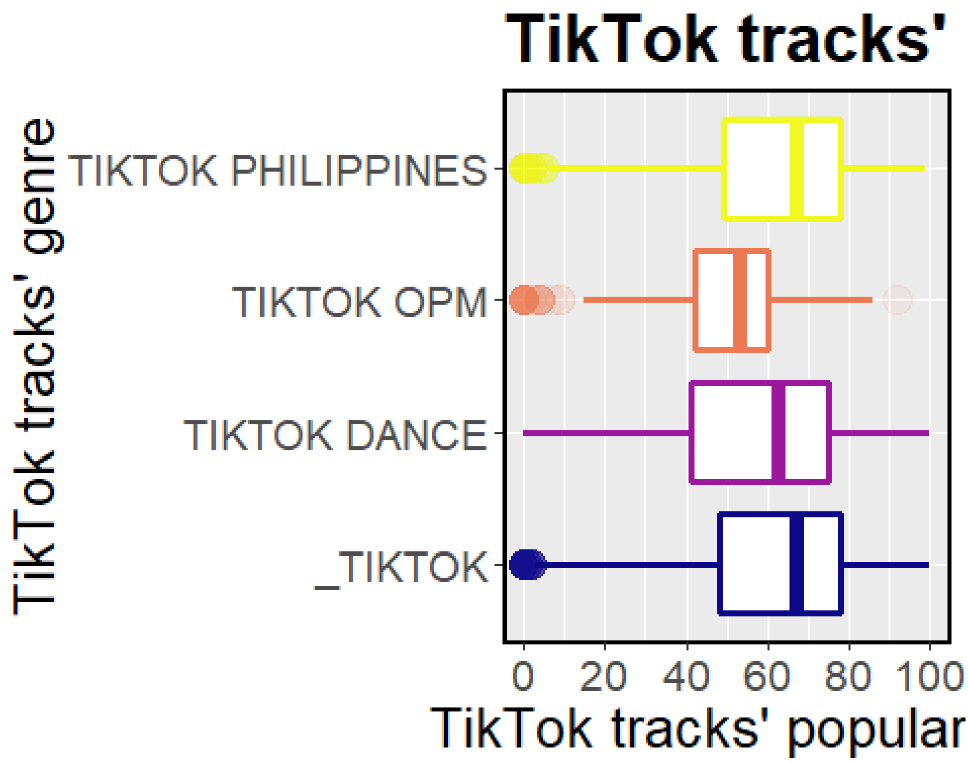
###Looking at distributions of categorical features Now that we explored the continuous variables, we aim to explore categorical variables. Is there any significant difference in TikTok tracks's popularity for TikTok tracks in different genres? Is there any significant difference in TikTok tracks's popularity for TikTok tracks with different modes? Are there different popularity distributions for different genres? Does median TikTok popularity changes for tracks with different genres? Can we see a different variability in popularity for different TikTok genres or/and tracks with different modes? To answer these questions, we plot popularity across levels of two categorical variables (Mode and Genre). The plots indicate that the TikTok tracks' popularity is symmetrically distributed among tracks with major and minor mode, and there aren't any outliers. The boxplot also shows that there aren't any outliers in TikTok dance genre. There are more outliers in TikTokk Philippines than other two genres though.

```
TikTok.data %>%
  ggplot(aes(x = mode_f, y = popularity, color = mode_f)) +
  geom_boxplot(size = 1.3,
               outlier.alpha = 1/15,
               outlier.size = 5) +
  scale_y_continuous(breaks = seq(0,100,20)) +
  scale_color_viridis_d(option = "plasma") +
  xlab("TikTok tracks' mode") +
```

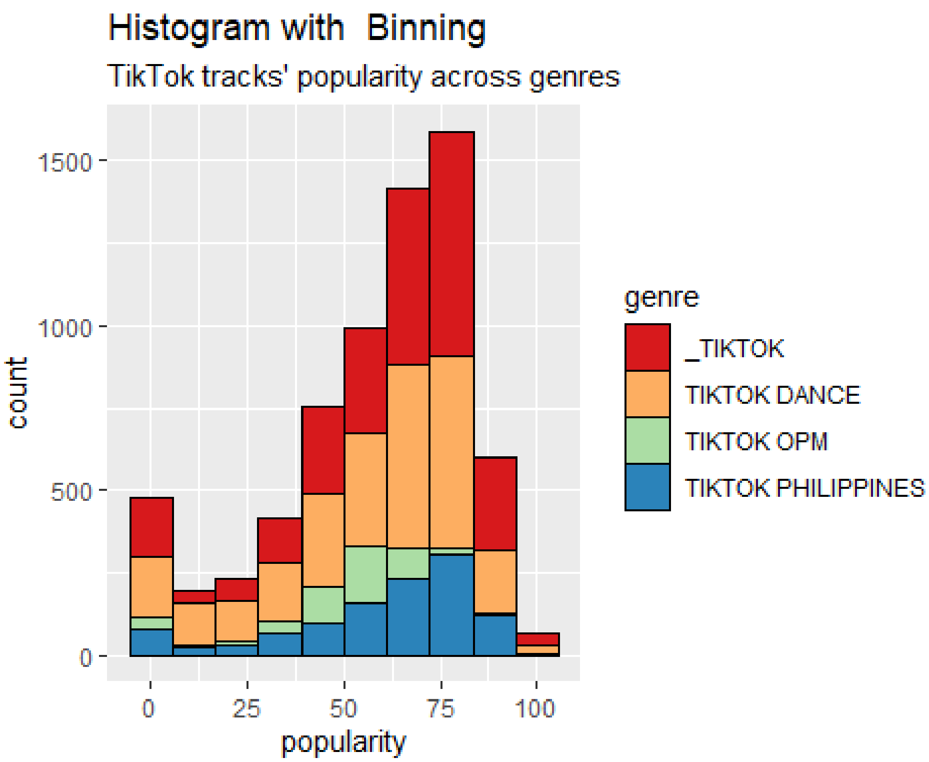
```
  ylab("TikTok tracks' popularity") +
  ggtitle("TikTok tracks' popularity VS mode - boxplot") +
  coord_flip() +
  theme(axis.text = element_text(size = 16),
        axis.title = element_text(size = 20),
        plot.title = element_text(size = 25, face = "bold"),
        panel.border = element_rect(color = "black", fill = NA, size = 1.2),
        legend.position = "none")
```



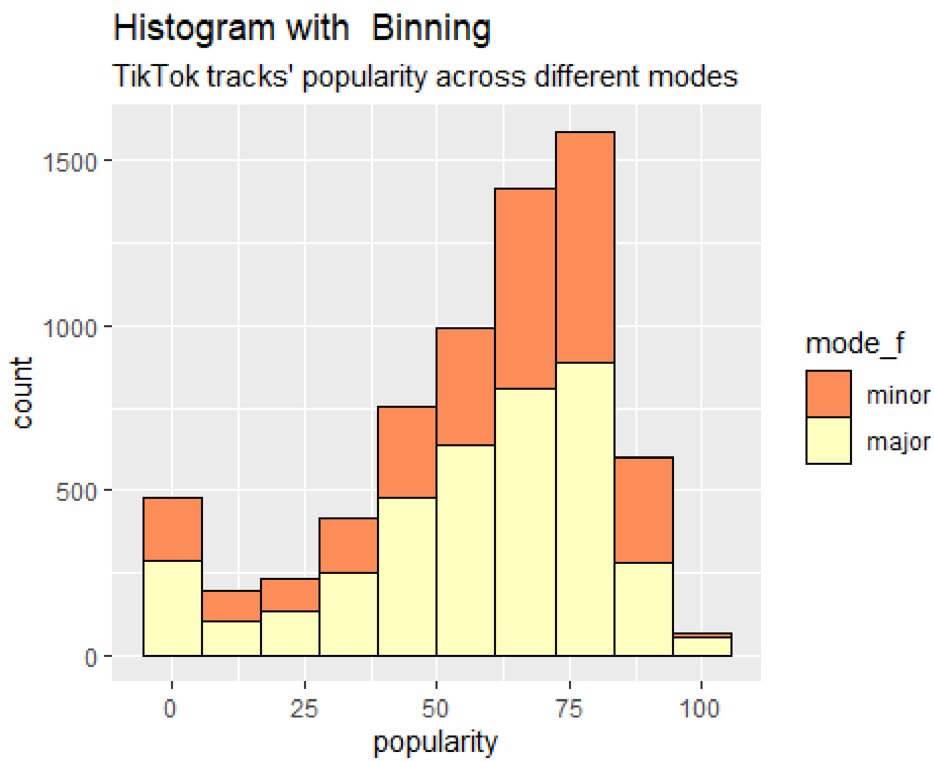

```
###

TikTok.data %>%
  ggplot(aes(x = genre, y = popularity, color = genre)) +
  geom_boxplot(size = 1.3,
               outlier.alpha = 1/15,
               outlier.size = 5) +
  scale_y_continuous(breaks = seq(0,100,20)) +
  scale_color_viridis_d(option = "plasma") +
  xlab("TikTok tracks' genre") +
  ylab("TikTok tracks' popularity") +
  ggtitle("TikTok tracks' popularity VS genre - boxplot") +
  coord_flip() +
  theme(axis.text = element_text(size = 16),
        axis.title = element_text(size = 20),
        plot.title = element_text(size = 25, face = "bold"),
        panel.border = element_rect(color = "black", fill = NA, size = 1.2),
        legend.position = "none")
```




```
#visualization of dependent variable (popularity) across levels of categorica
l variabes (Mode and genre):
g <- ggplot(TikTok.data, aes(popularity)) + scale_fill_brewer(palette = "Spec
tral")

g + geom_histogram(aes(fill=genre),
                   bins=10 ,
                   col="black",
                   size=.1) +  # change binwidth
  labs(title="Histogram with  Binning",
       subtitle="TikTok tracks' popularity across genres")
```




```
d <- ggplot(TikTok.data, aes(popularity)) + scale_fill_brewer(palette = "Spec
tral")

d + geom_histogram(aes(fill=mode_f),
                   bins=10 ,
                   col="black",
                   size=.1) +  # change binwidth
  labs(title="Histogram with  Binning",
       subtitle="TikTok tracks' popularity across different modes")
```



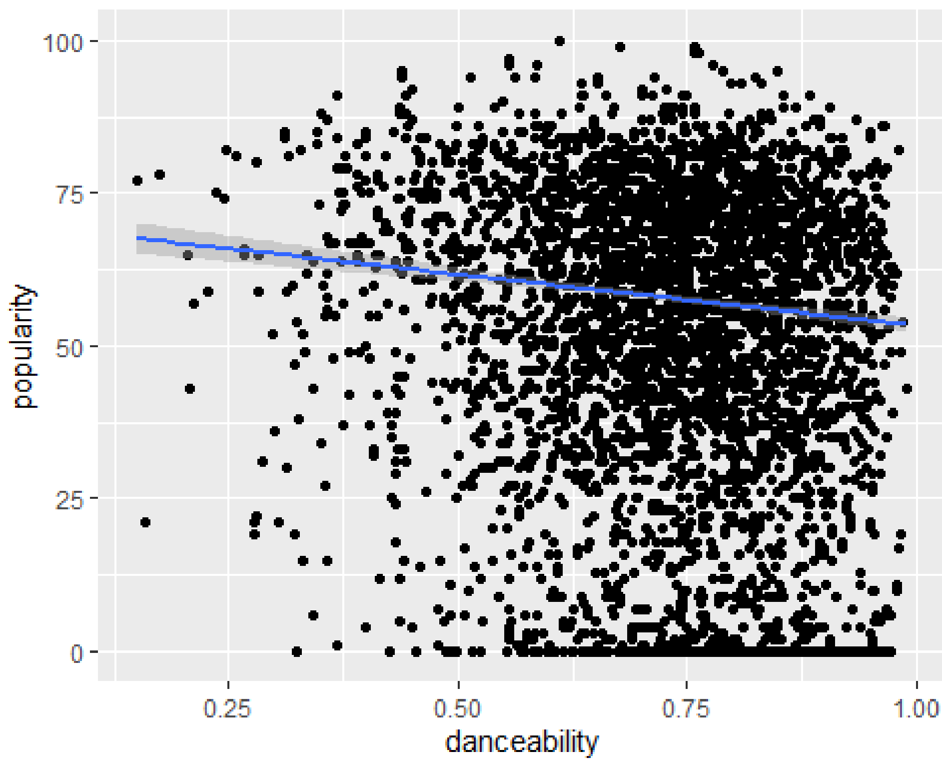
Now that we have some understanding of the data, we fit the initial model in order to refine it later.

```
m1 <- lm(popularity~danceability + loudness + instrumentalness + tempo + vale
nce + energy)
```

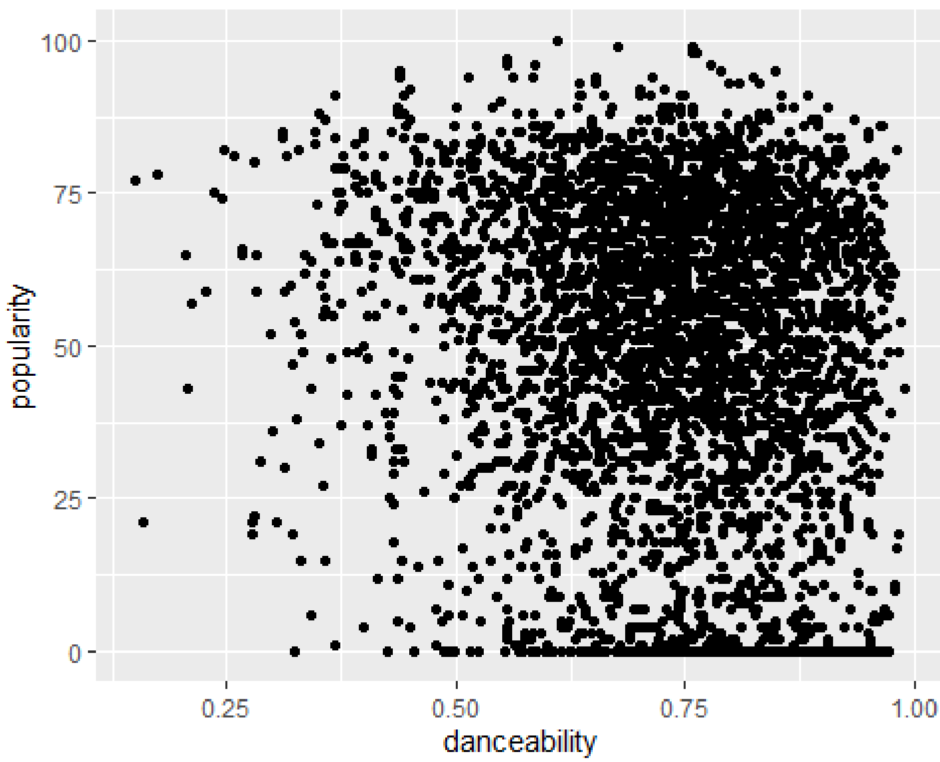
Looking at the relationship between numeric and dependent variable

We aim to explore whether selected variables are somehow connected. In other words, we want to understand if there is just some random connection between variables or we can see some patterns when we compare variables. From this point, our main focus is to check how TikTok tracks' popularity is related to other variables. We also use the "" method to plot the best model for the relation between the dependent variabel and the target variabels based on the algorithm choice. In addition,To find a model that is the best model for our data, we transform our variable using log og transformation. We use linear log, log lienar, and then log log as we hope to obtain the linear relation with transformed variables.For some variables, algorithm chooses the non-linear model for our data.

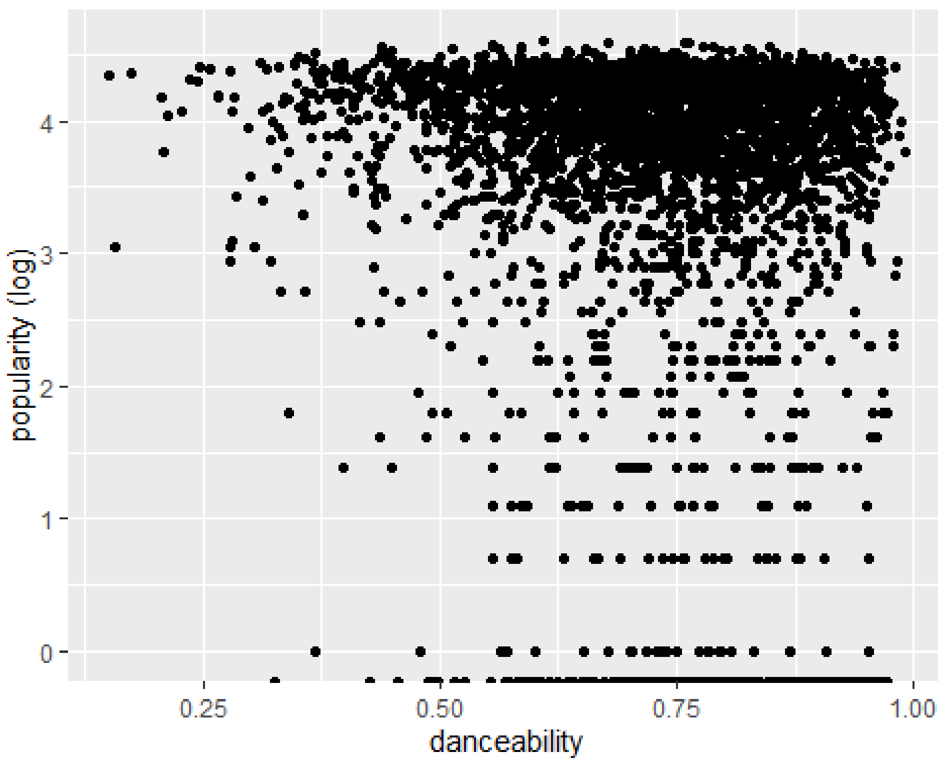
```
# use geom_smooth - for adding regression model
TikTok.data %>%
  ggplot(aes(x = danceability, y = popularity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = 'y ~ x')
```



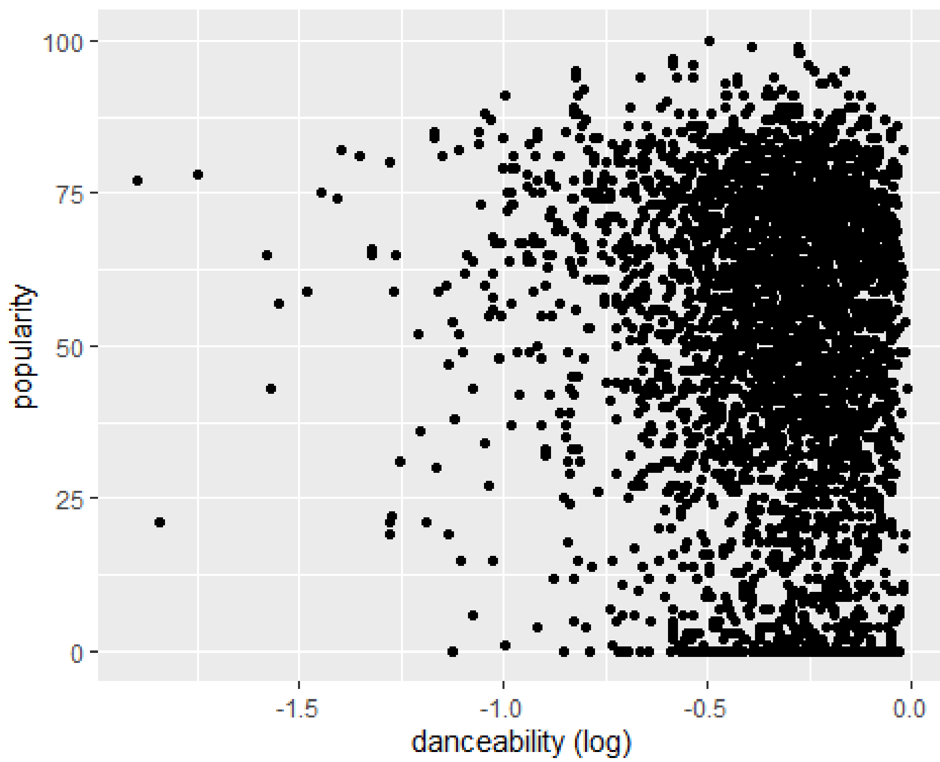

```
#
# without confidence intervals around smoothed line; no standard error (se)
TikTok.data %>%
  ggplot(aes(x = danceability, y = popularity)) +
  geom_point() +
  geom_smooth(method = "",  se = TRUE, color = "red")

## `geom_smooth()` using formula 'y ~ x'

## Warning: Computation failed in `stat_smooth()`:
## invalid first argument
```



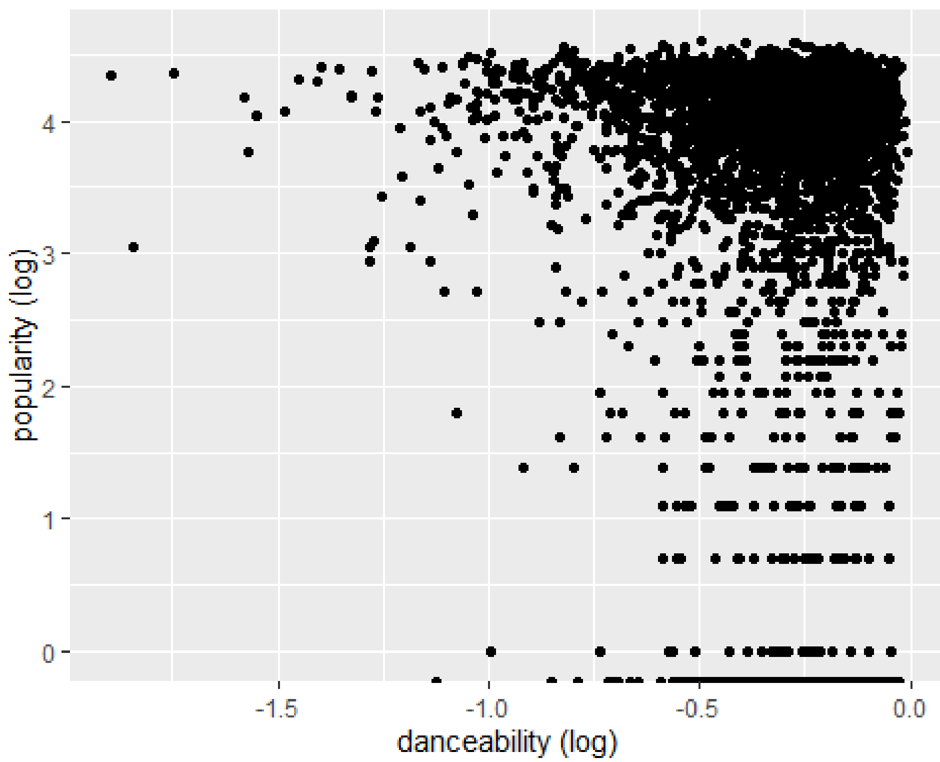

```
###Algorithm chooses this model for our data.
###


# Transforming variables
#   try to use logarithmic transformation on popularity and/or on danceabilit
y
#

# transformation manually (natural logarithm ln=log):
TikTok.data <- TikTok.data %>%
  mutate(`popularity (log)` = log(popularity),
         `danceability (log)` = log(danceability))

# log popularity VS danceability - not linear relation :(
TikTok.data %>%
  ggplot(aes(x = danceability, y = `popularity (log)`)) +
  geom_point()
```



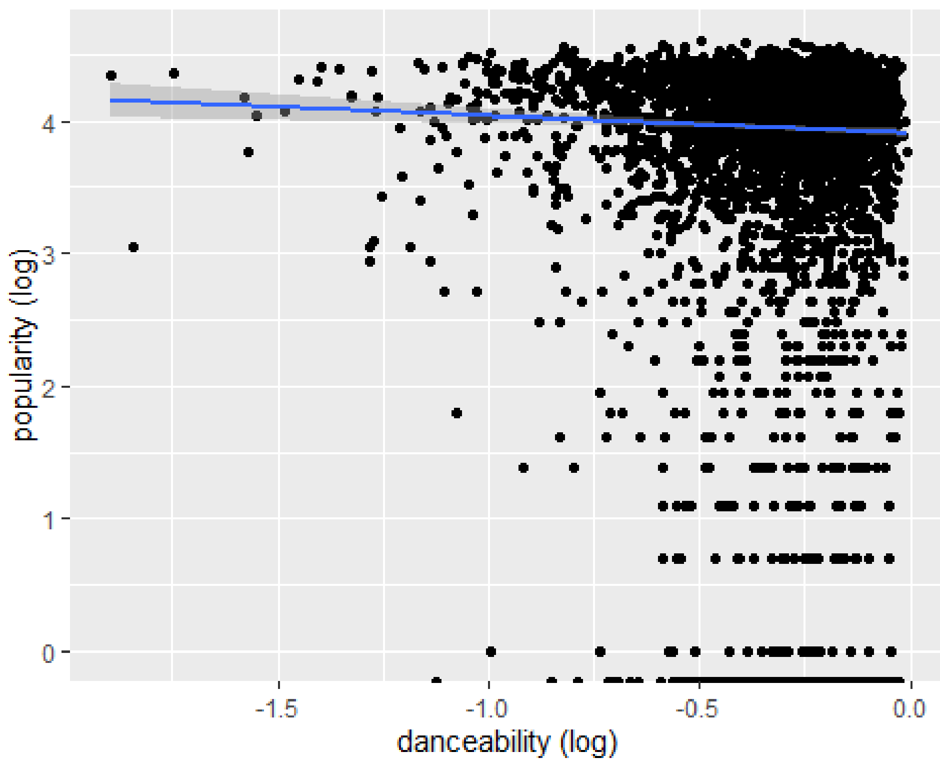

```
# popularity VS log danceability - not linear relation :(
TikTok.data %>%
  ggplot(aes(x = `danceability (log)`, y = popularity)) +
  geom_point()
```



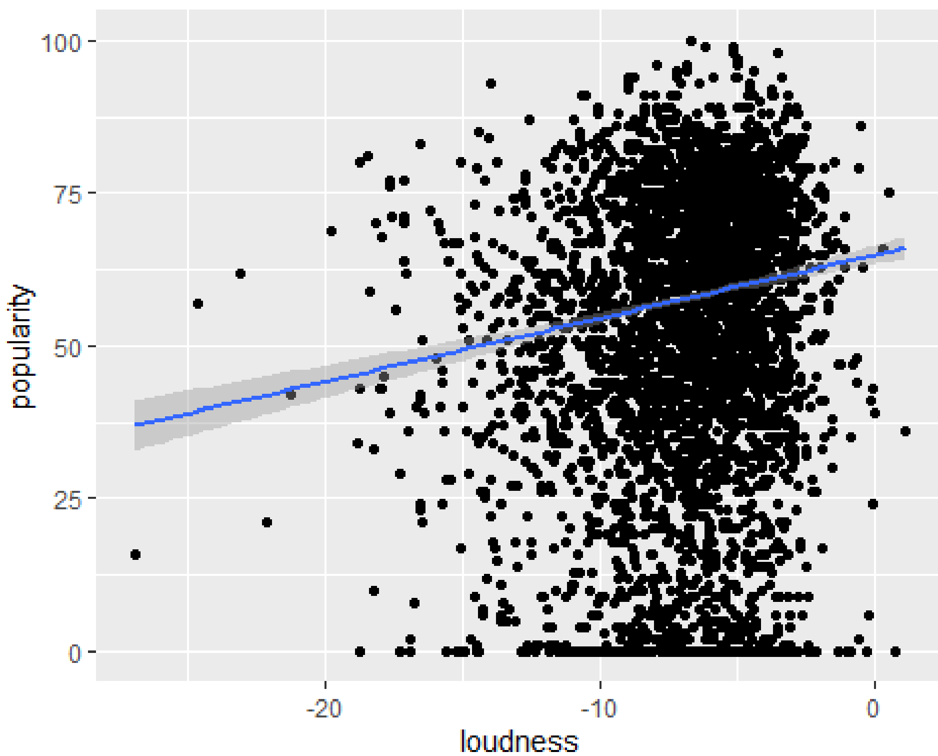

```
# log popularity VS log danceability - not linear relation:(
TikTok.data %>%
  ggplot(aes(x = `danceability (log)`, y = `popularity (log)`)) +
  geom_point()
```



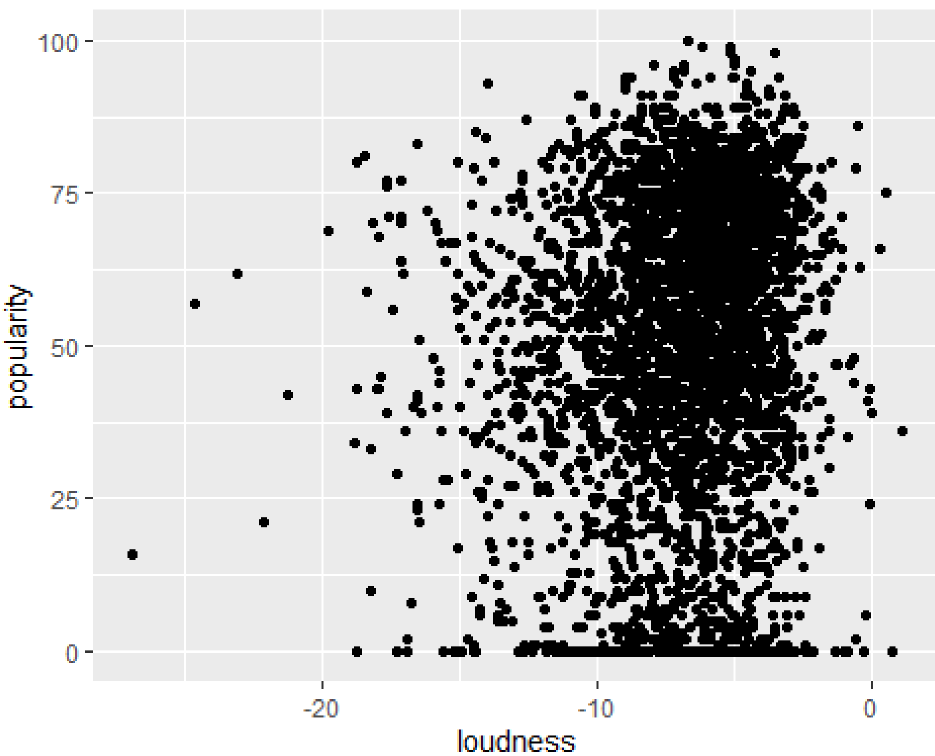

```
# let's fit a model on this transformed variables
TikTok.data %>%
  ggplot(aes(x = `danceability (log)`, y = `popularity (log)`)) +
  geom_point() +
  geom_smooth(method = "lm")

## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 276 rows containing non-finite values (stat_smooth).
```



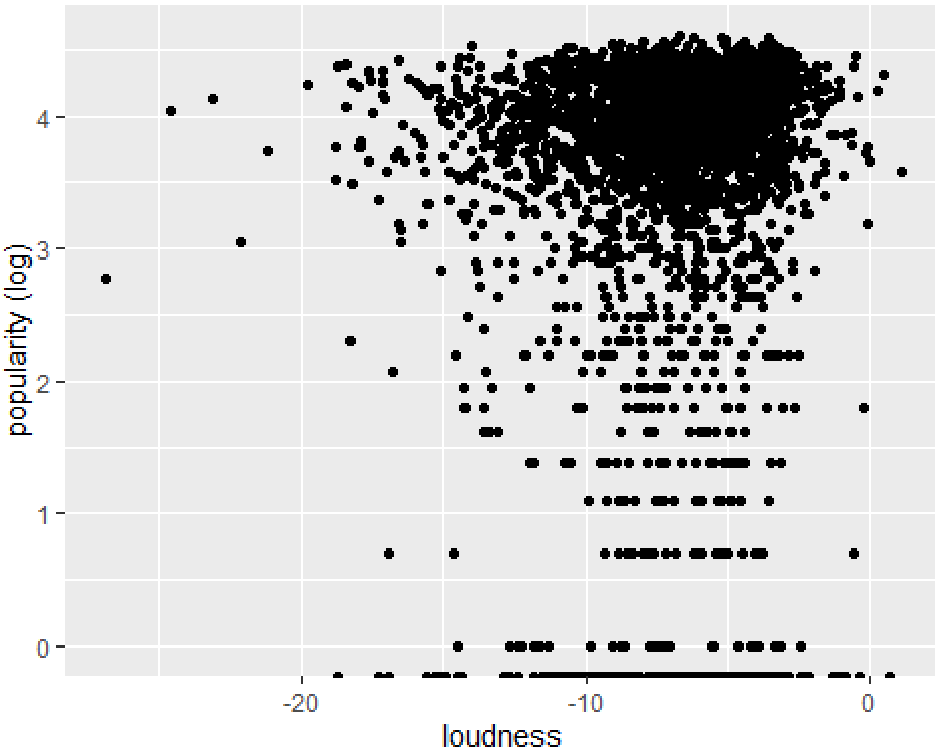

```
# use geom_smooth - for adding regression model
TikTok.data %>%
  ggplot(aes(x = loudness, y = popularity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = 'y ~ x')
```



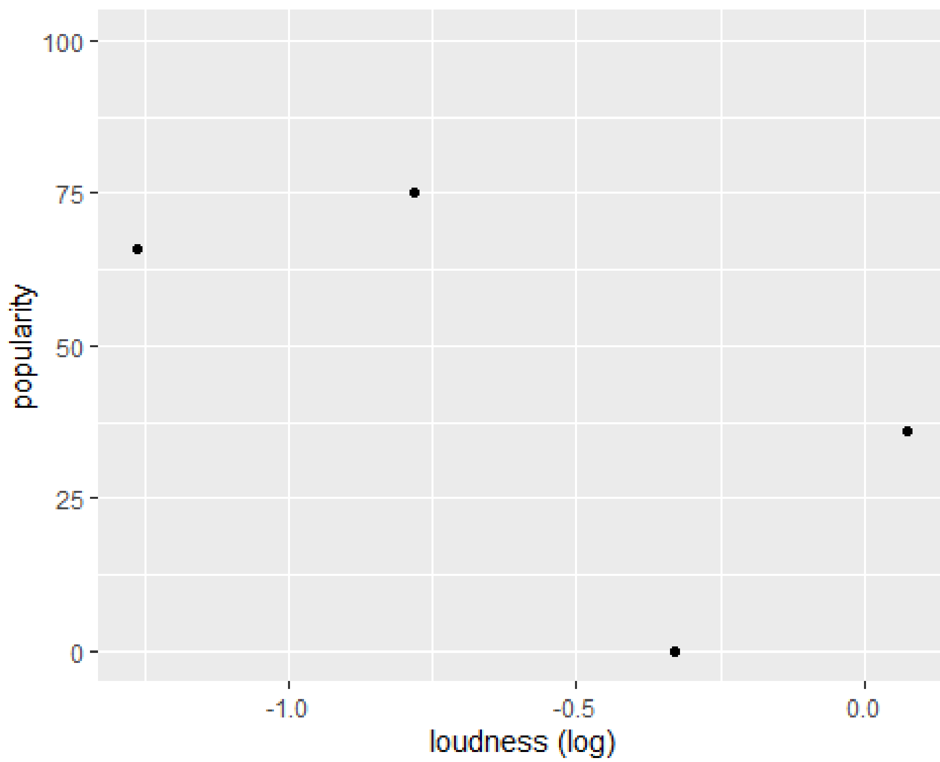

```
# without confidence intervals around smoothed line; no standard error (se)
TikTok.data %>%
  ggplot(aes(x = loudness, y = popularity)) +
  geom_point() +
  geom_smooth(method = "",  se = TRUE, color = "red")

## `geom_smooth()` using formula 'y ~ x'

## Warning: Computation failed in `stat_smooth()`:
## invalid first argument
```



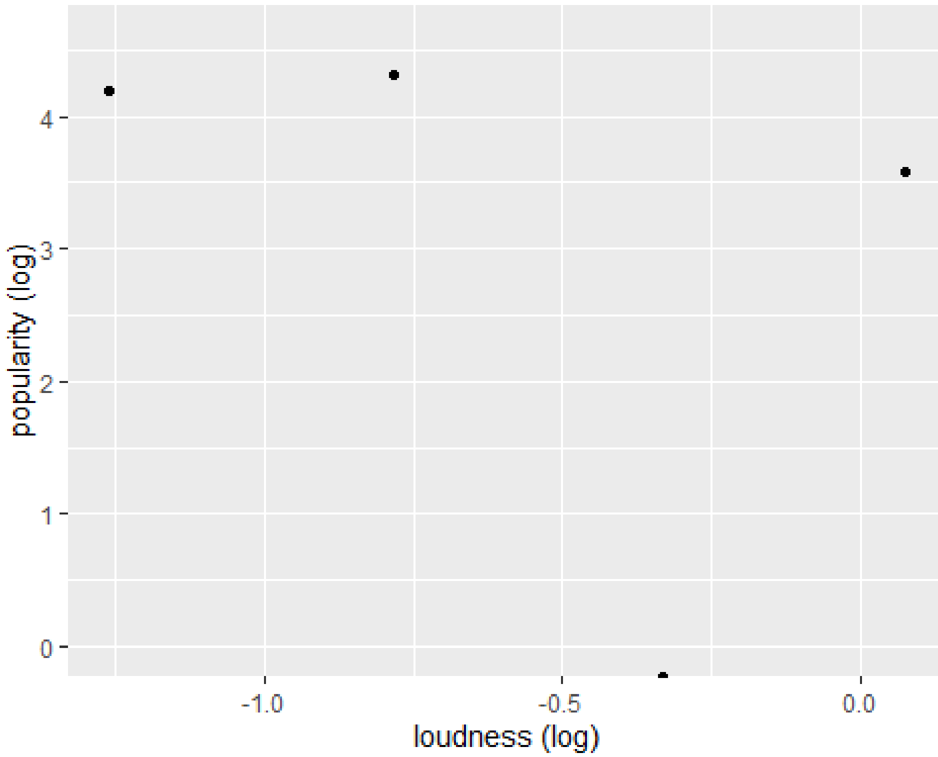

```
###Algorithm chooses the model non-linear model for our data.
###To find a model that is the best model for our data, we transform our vari
able using log log transformation. we use linear log, log lienar, and then lo
g log.


# Transforming variables
#   try to use logarithmic transformation on popularity and/or on loudness
#   maybe we can obtain linear relation with transformed variables

# transformation manually (natural logarithm ln=log):
TikTok.data <- TikTok.data %>%
  mutate(`popularity (log)` = log(popularity),
         `loudness (log)` = log(loudness))

## Warning in log(loudness): NaNs produced

# log popularity VS loudness - not linear relation :(
TikTok.data %>%
  ggplot(aes(x = loudness, y = `popularity (log)`)) +
  geom_point()
```



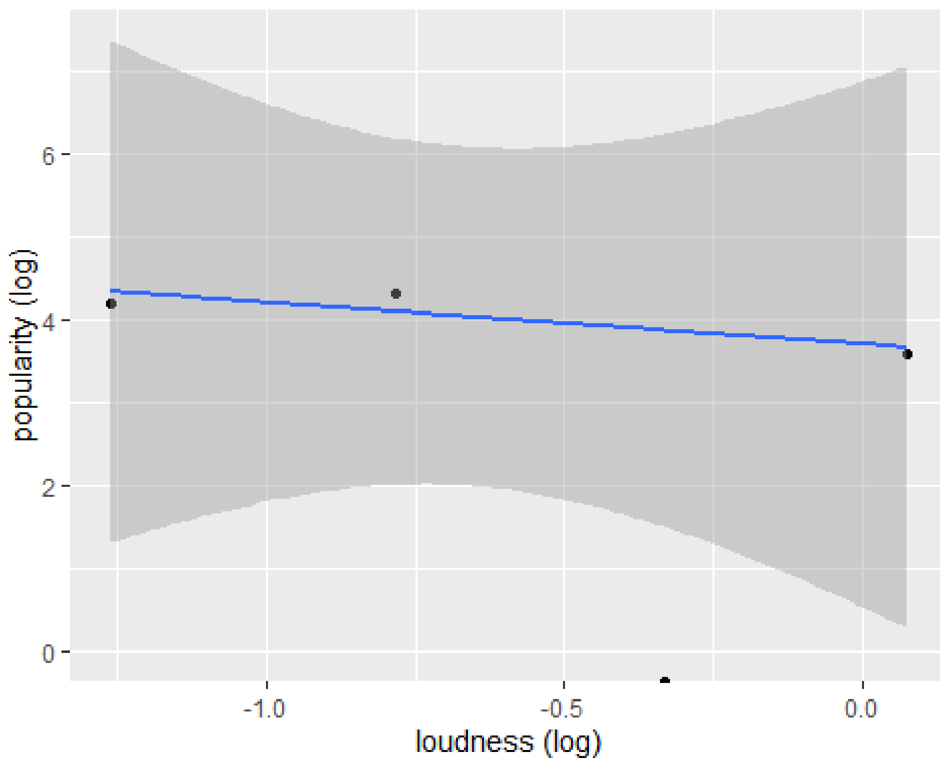

```r
# popularity VS log loudness - not linear relation :(
TikTok.data %>%
  ggplot(aes(x = `loudness (log)`, y = popularity)) +
  geom_point()

## Warning: Removed 6742 rows containing missing values (geom_point).
```



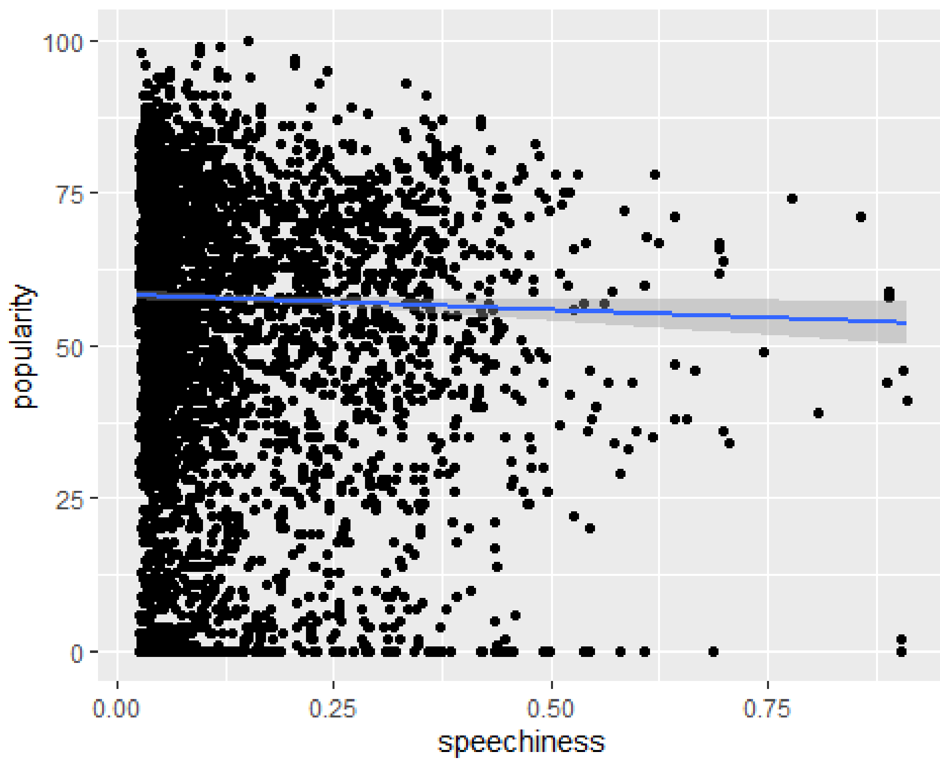

```
# log popularity VS log loudness - not linear relation:(
TikTok.data %>%
  ggplot(aes(x = `loudness (log)`, y = `popularity (log)`)) +
  geom_point()

## Warning: Removed 6742 rows containing missing values (geom_point).
```



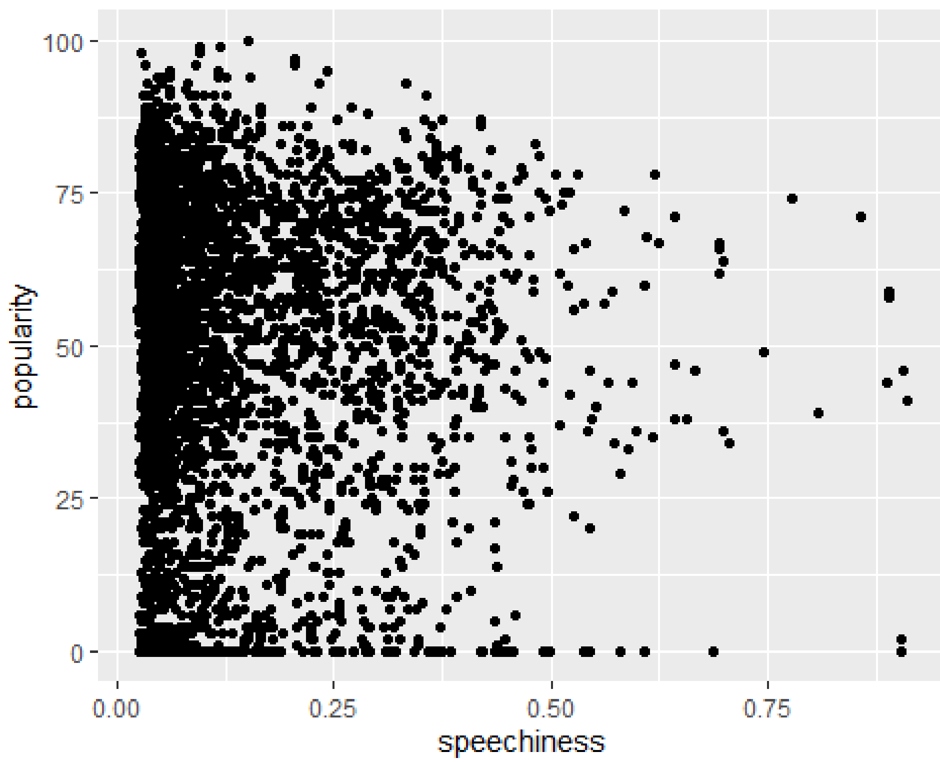

```r
# let's fit a model on this transformed variables
TikTok.data %>%
  ggplot(aes(x = `loudness (log)`, y = `popularity (log)`)) +
  geom_point() +
  geom_smooth(method = "lm")

## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 6743 rows containing non-finite values (stat_smooth).

## Warning: Removed 6742 rows containing missing values (geom_point).
```



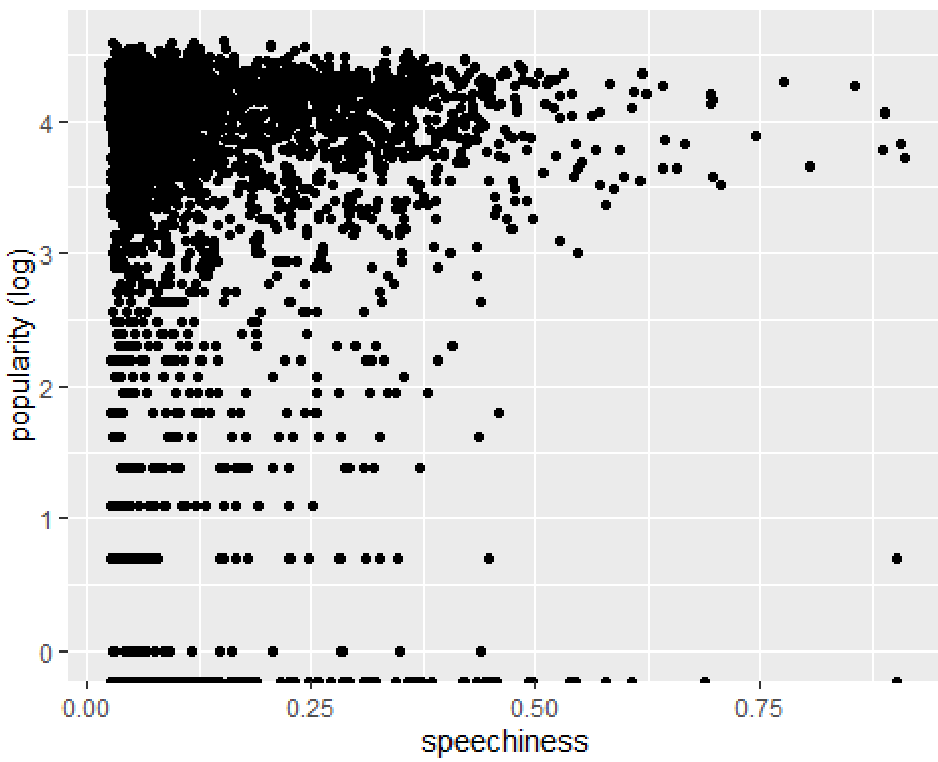

```
# use geom_smooth - for adding regression model
TikTok.data %>%
  ggplot(aes(x = speechiness, y = popularity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = 'y ~ x')
```



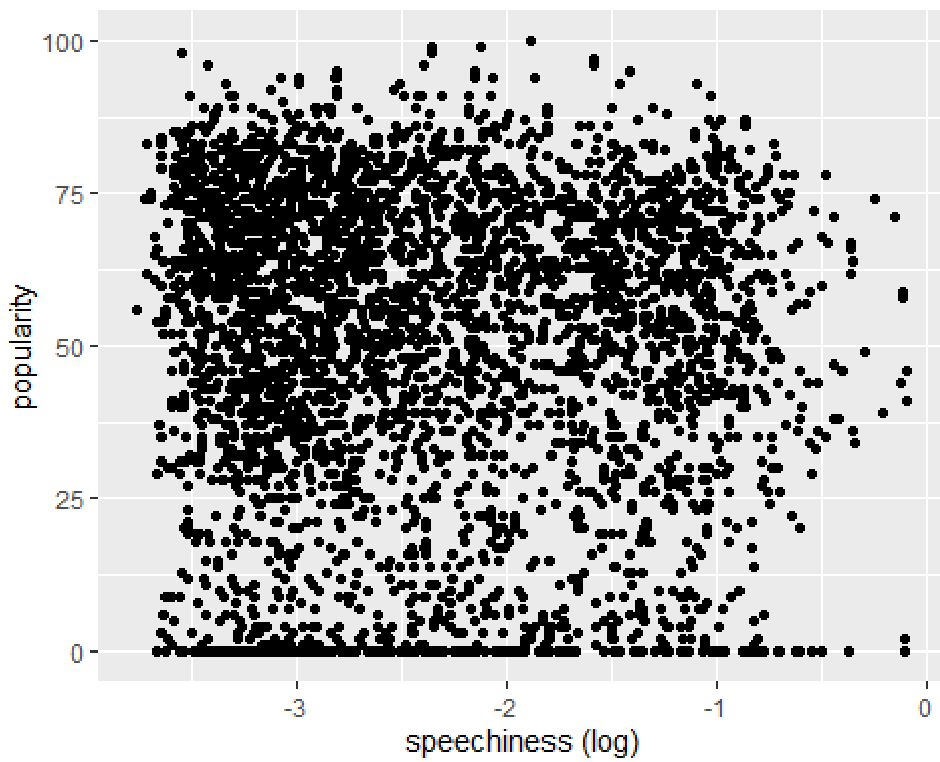

```r
# without confidence intervals around smoothed line; no standard error (se)
TikTok.data %>%
  ggplot(aes(x = speechiness, y = popularity)) +
  geom_point() +
  geom_smooth(method = "",  se = TRUE, color = "red")

## `geom_smooth()` using formula 'y ~ x'

## Warning: Computation failed in `stat_smooth()`:
## invalid first argument
```



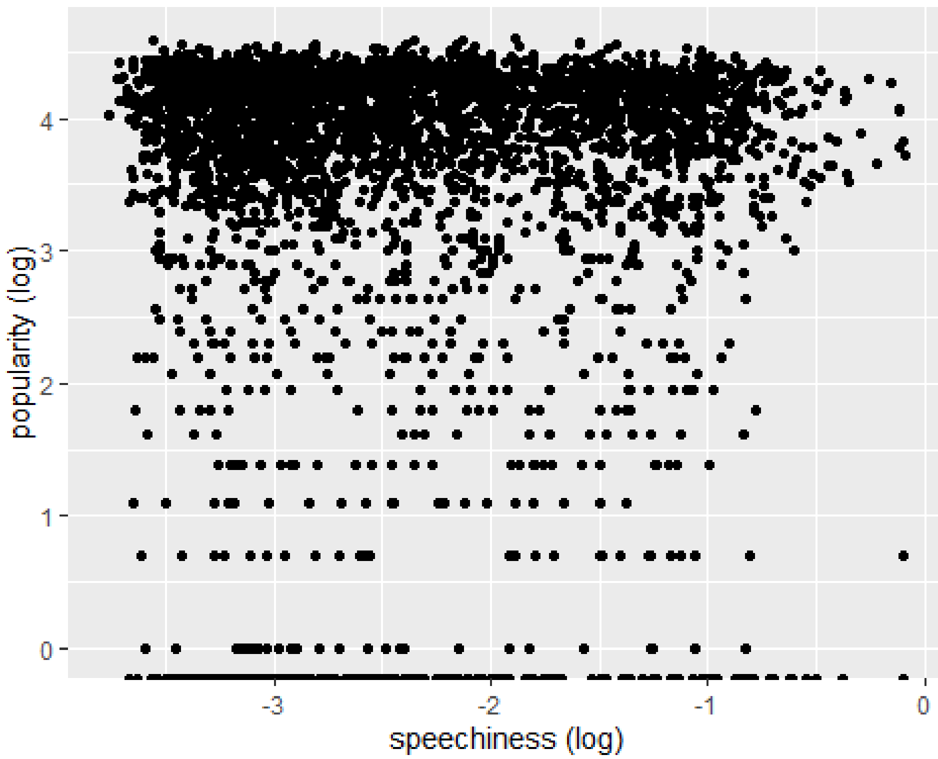

```
###Algorithm chooses the  non-linear model for our data.
###To find a model that is the best model for our data, we transform our vari
able using log log transformation. we use linear log, log lienar, and then lo
g log.


# Transforming variables
#   try to use logarithmic transformation on popularity and/or on speechiness
#   maybe we can obtain linear relation with transformed variables

# transformation manually (natural logarithm ln=log):
TikTok.data <- TikTok.data %>%
  mutate(`popularity (log)` = log(popularity),
         `speechiness (log)` = log(speechiness))

# log popularity VS speechiness - not linear relation :(
TikTok.data %>%
  ggplot(aes(x = speechiness, y = `popularity (log)`)) +
  geom_point()
```



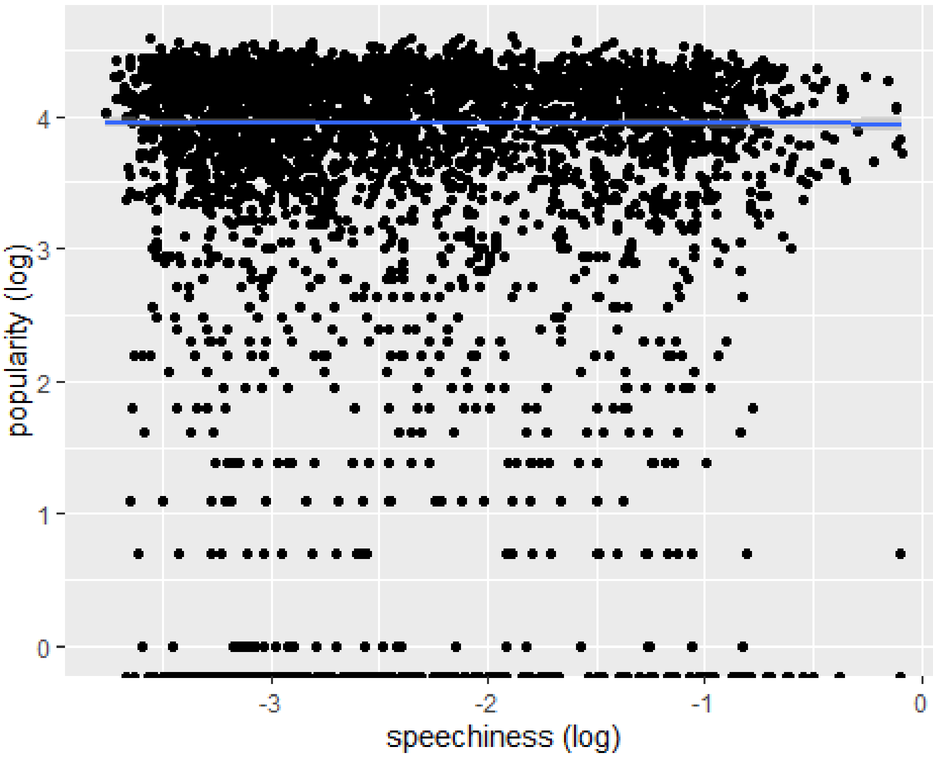

```
# popularity VS log speechiness - not linear relation :(
TikTok.data %>%
  ggplot(aes(x = `speechiness (log)`, y = popularity)) +
  geom_point()
```



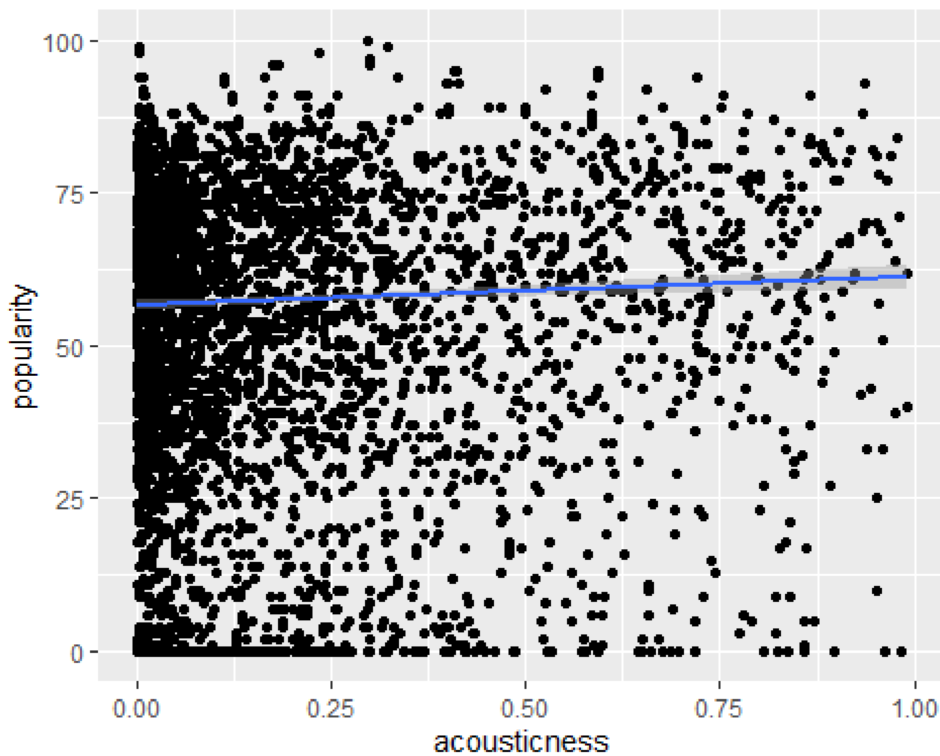

```r
# log popularity VS log speechiness - not linear relation:(
TikTok.data %>%
  ggplot(aes(x = `speechiness (log)`, y = `popularity (log)`)) +
  geom_point()
```



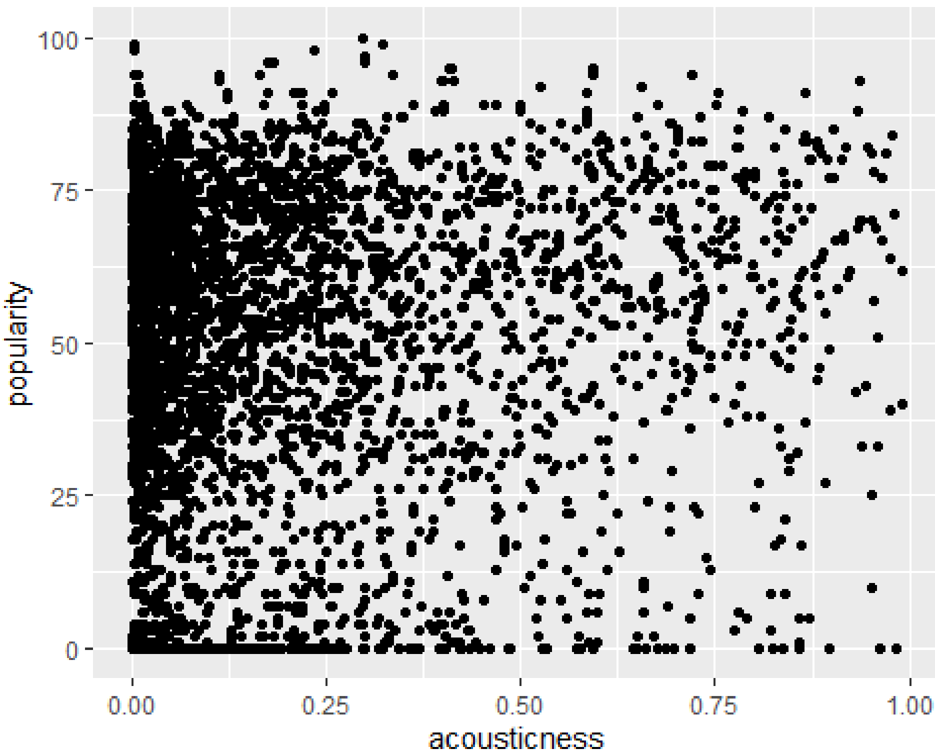

```
# let's fit a model on this transformed variables
TikTok.data %>%
  ggplot(aes(x = `speechiness (log)`, y = `popularity (log)`)) +
  geom_point() +
  geom_smooth(method = "lm")

## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 276 rows containing non-finite values (stat_smooth).
```



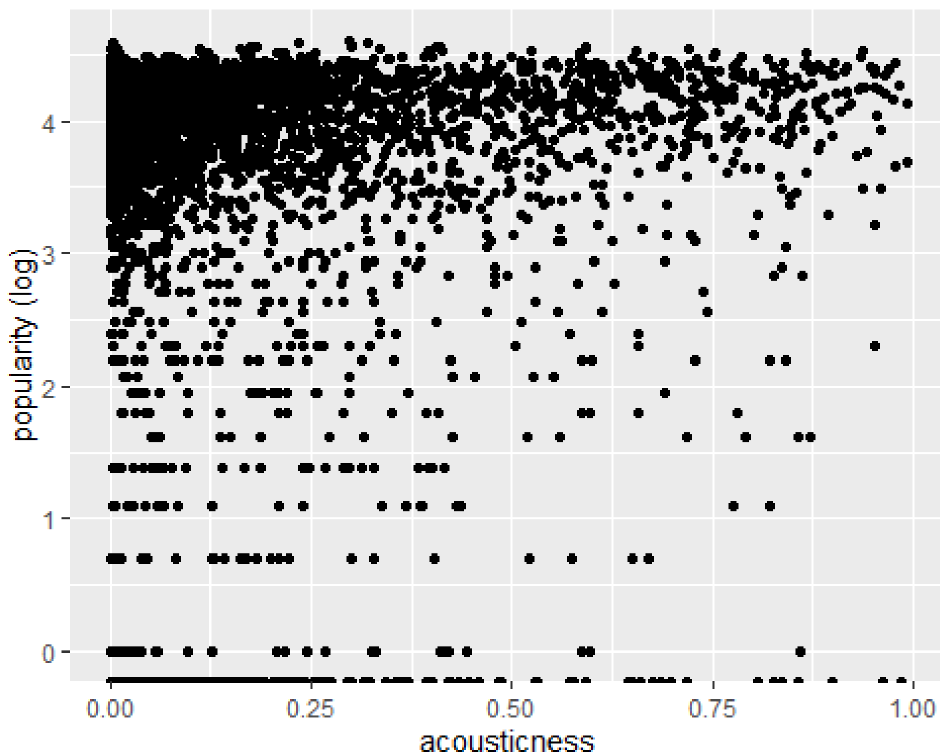

```
###acousticness


# use geom_smooth - for adding regression model
TikTok.data %>%
  ggplot(aes(x = acousticness, y = popularity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = 'y ~ x')
```



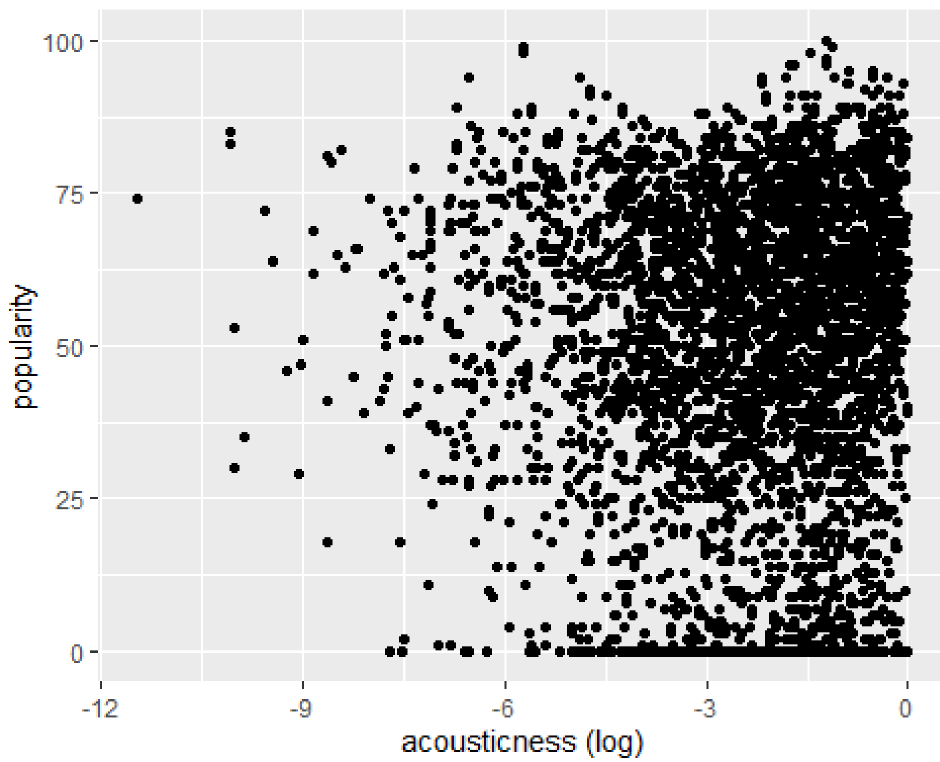

```
# without confidence intervals around smoothed line; no standard error (se)
TikTok.data %>%
  ggplot(aes(x = acousticness, y = popularity)) +
  geom_point() +
  geom_smooth(method = "",  se = TRUE, color = "red")

## `geom_smooth()` using formula 'y ~ x'

## Warning: Computation failed in `stat_smooth()`:
## invalid first argument
```



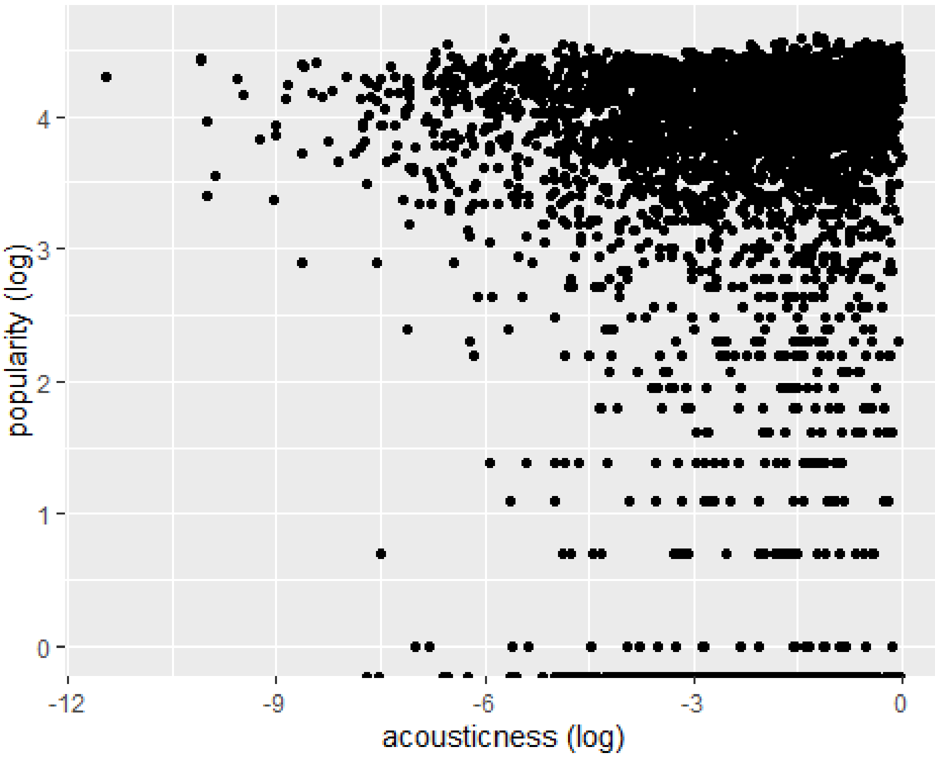

```
###Algorithm chooses the  non-linear model for our data.
###To find a model that is the best model for our data, we transform our vari
able using log log transformation. we use linear log, log lienar, and then lo
g log.


# Transforming variables
#   try to use logarithmic transformation on popularity and/or on acousticnes
s
#   maybe we can obtain linear relation with transformed variables

# transformation manually (natural logarithm ln=log):
TikTok.data <- TikTok.data %>%
  mutate(`popularity (log)` = log(popularity),
         `acousticness (log)` = log(acousticness))

# log popularity VS acousticness - not linear relation :(
TikTok.data %>%
  ggplot(aes(x = acousticness, y = `popularity (log)`)) +
  geom_point()
```



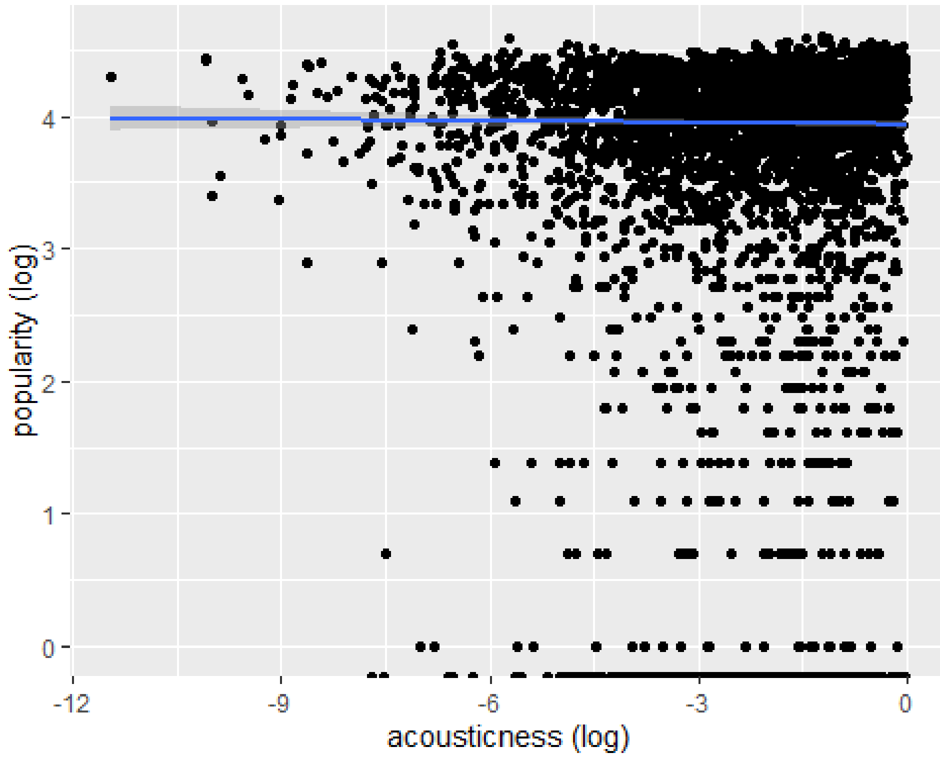

```
# popularity VS log acousticness - not linear relation :(
TikTok.data %>%
  ggplot(aes(x = `acousticness (log)`, y = popularity)) +
  geom_point()
```



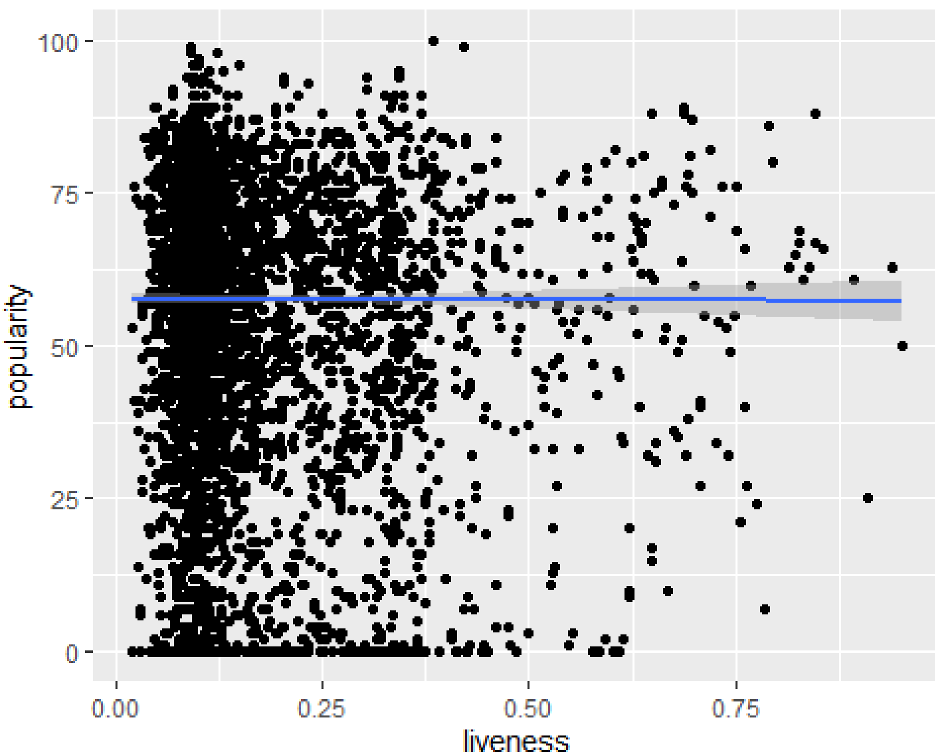

```r
# log popularity VS log acousticness - not linear relation:(
TikTok.data %>%
  ggplot(aes(x = `acousticness (log)`, y = `popularity (log)`)) +
  geom_point()
```



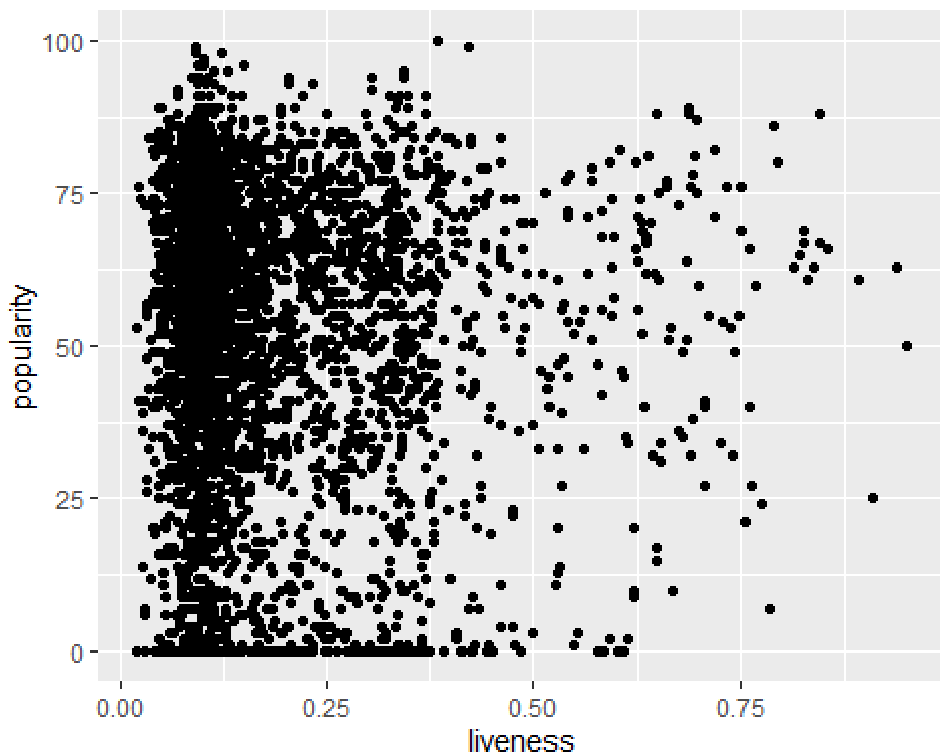

```
# let's fit a model on this transformed variables
TikTok.data %>%
  ggplot(aes(x = `acousticness (log)`, y = `popularity (log)`)) +
  geom_point() +
  geom_smooth(method = "lm")

## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 276 rows containing non-finite values (stat_smooth).
```



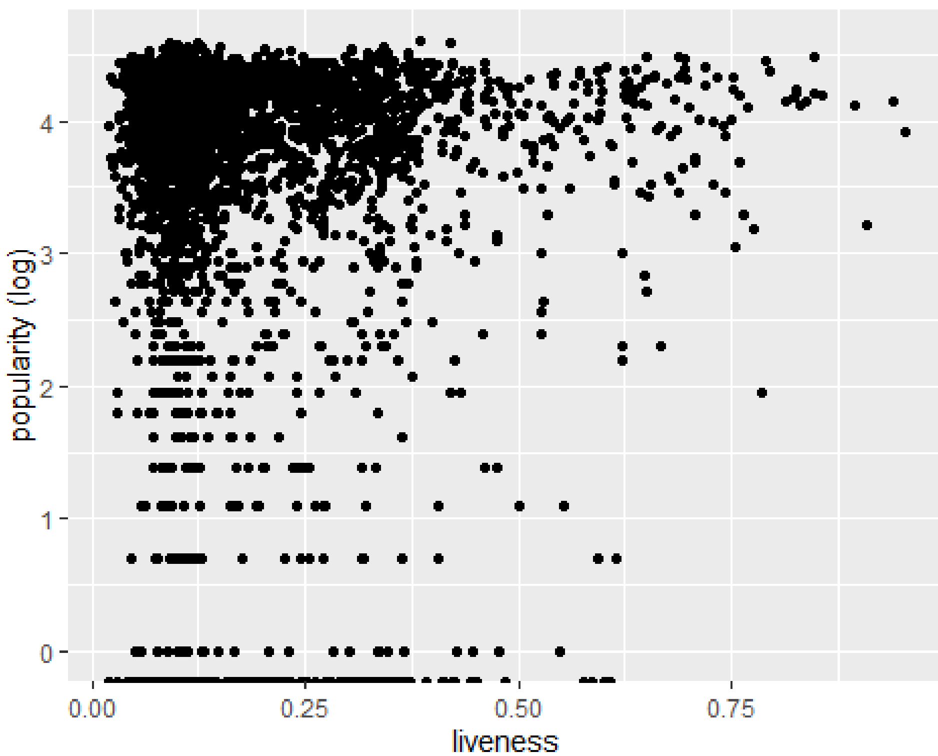

```
####liveness
# use geom_smooth - for adding regression model
TikTok.data %>%
  ggplot(aes(x = liveness, y = popularity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = 'y ~ x')
```



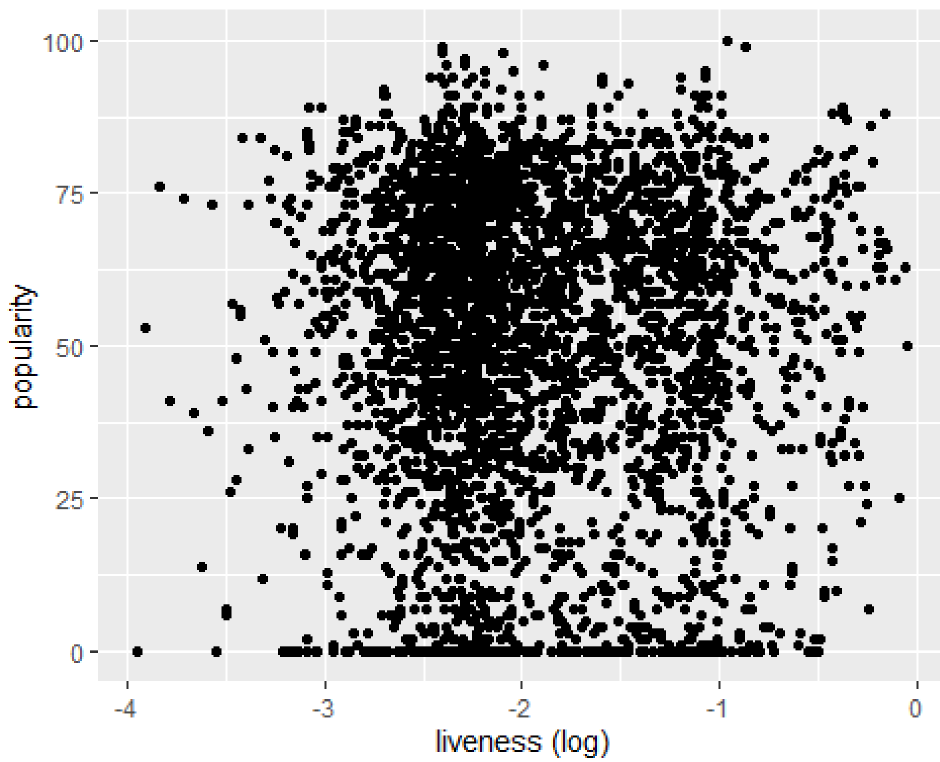

```
# without confidence intervals around smoothed line; no standard error (se)
TikTok.data %>%
  ggplot(aes(x = liveness, y = popularity)) +
  geom_point() +
  geom_smooth(method = "",  se = TRUE, color = "red")

## `geom_smooth()` using formula 'y ~ x'

## Warning: Computation failed in `stat_smooth()`:
## invalid first argument
```



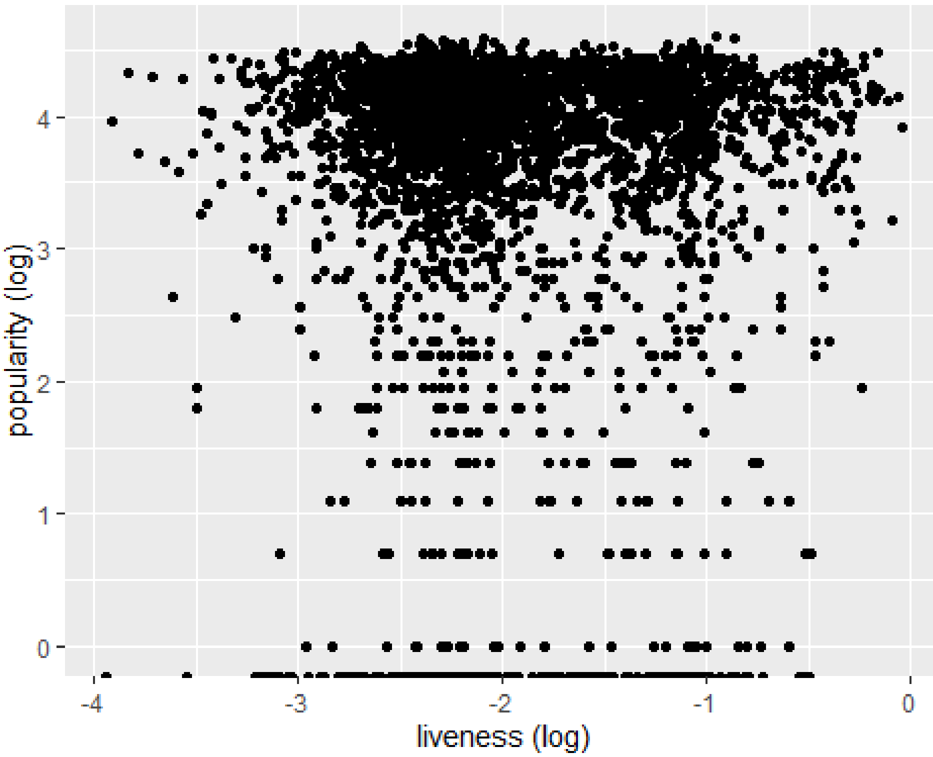

```
###Algorithm chooses the  non-linear model for our data.
###To find a model that is the best model for our data, we transform our vari
able using log log transformation. we use linear log, log lienar, and then lo
g log.


# Transforming variables
#   try to use logarithmic transformation on popularity and/or on liveness
#   maybe we can obtain linear relation with transformed variables

# transformation manually (natural logarithm ln=log):
TikTok.data <- TikTok.data %>%
  mutate(`popularity (log)` = log(popularity),
         `liveness (log)` = log(liveness))

# log popularity VS liveness - not linear relation :(
TikTok.data %>%
  ggplot(aes(x = liveness, y = `popularity (log)`)) +
  geom_point()
```



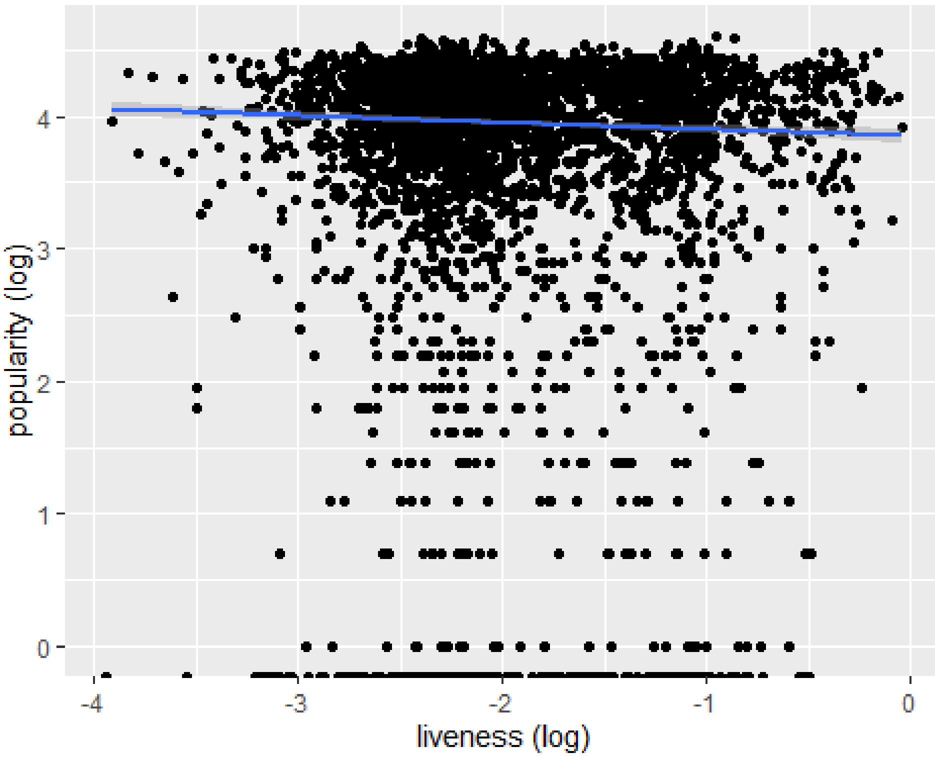

```r
# popularity VS log liveness - not linear relation :(
TikTok.data %>%
  ggplot(aes(x = `liveness (log)`, y = popularity)) +
  geom_point()
```



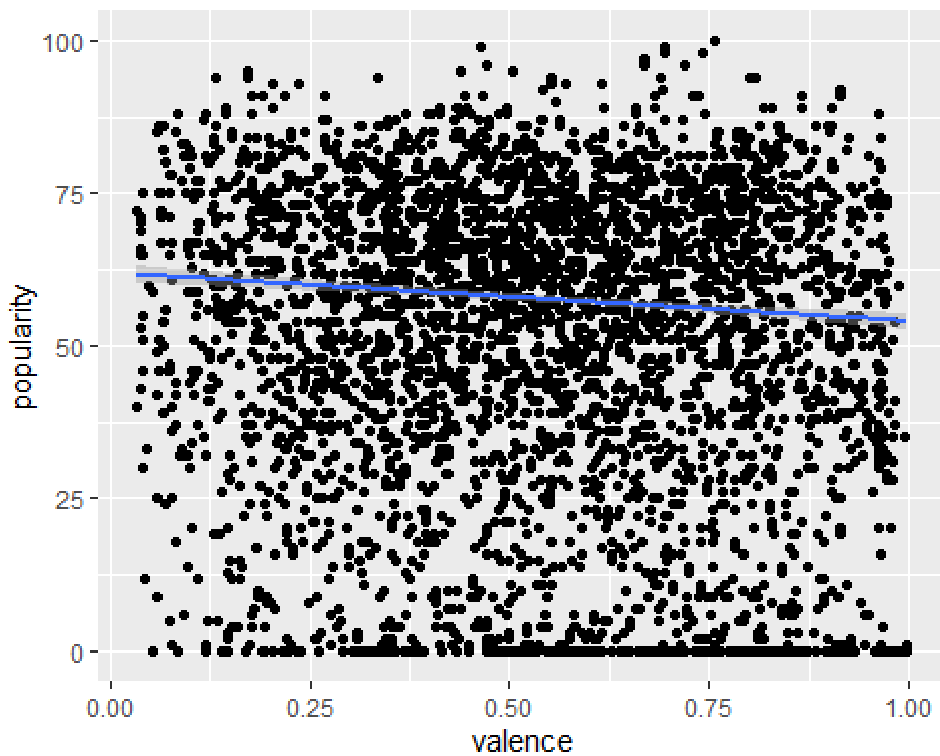

```
# log popularity VS log liveness - not linear relation:(
TikTok.data %>%
  ggplot(aes(x = `liveness (log)`, y = `popularity (log)`)) +
  geom_point()
```



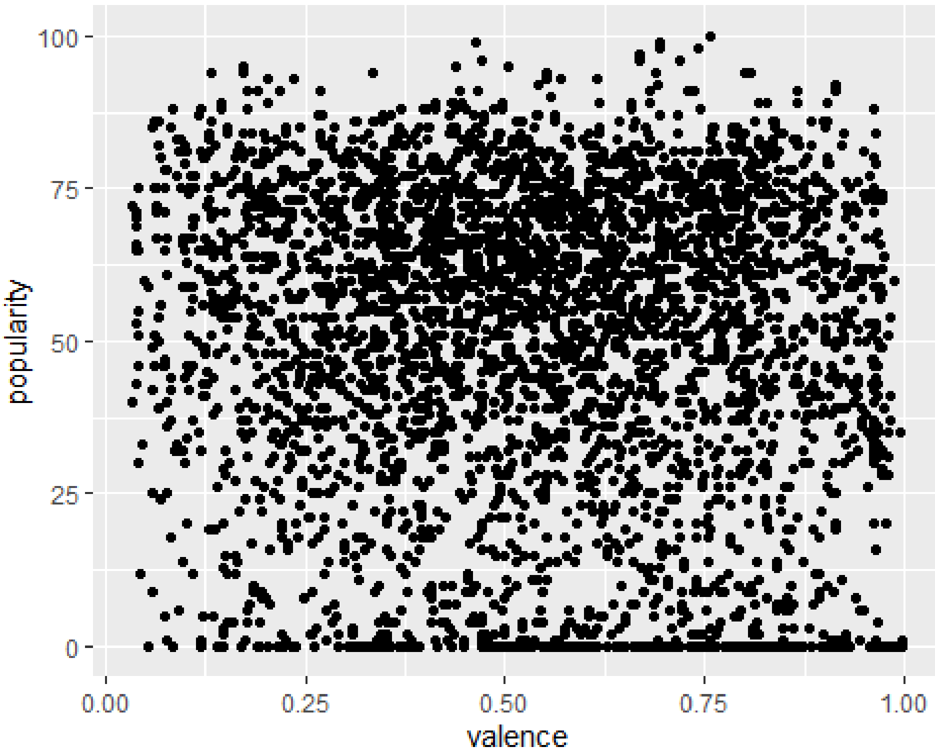

```r
# let's fit a model on this transformed variables
TikTok.data %>%
  ggplot(aes(x = `liveness (log)`, y = `popularity (log)`)) +
  geom_point() +
  geom_smooth(method = "lm")

## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 276 rows containing non-finite values (stat_smooth).
```



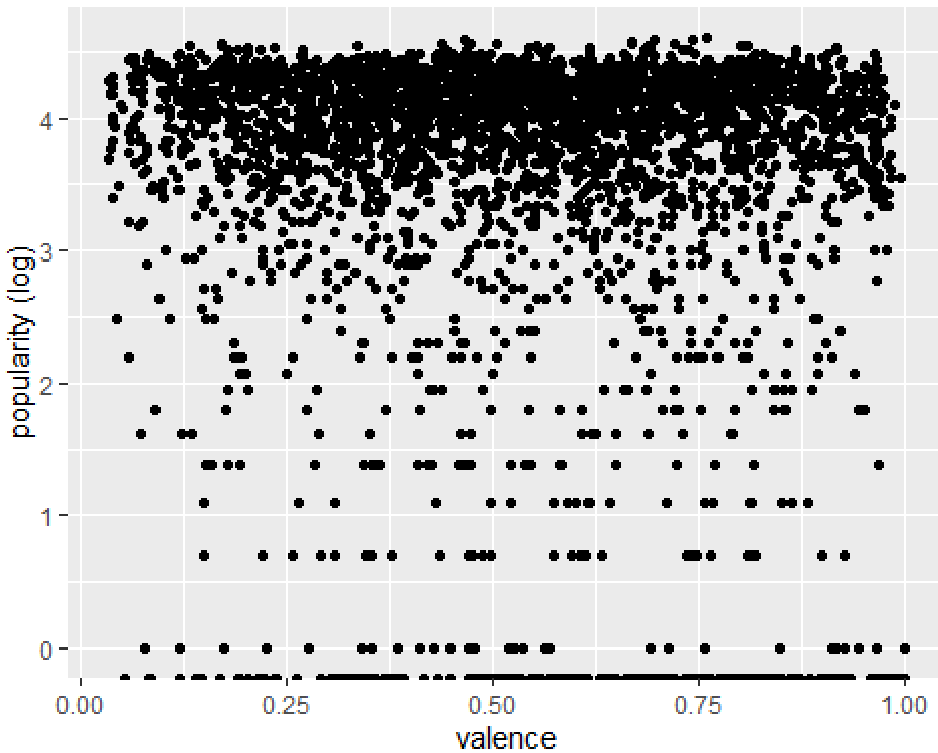

```r
####valence
# use geom_smooth - for adding regression model
TikTok.data %>%
  ggplot(aes(x = valence, y = popularity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = 'y ~ x')
```



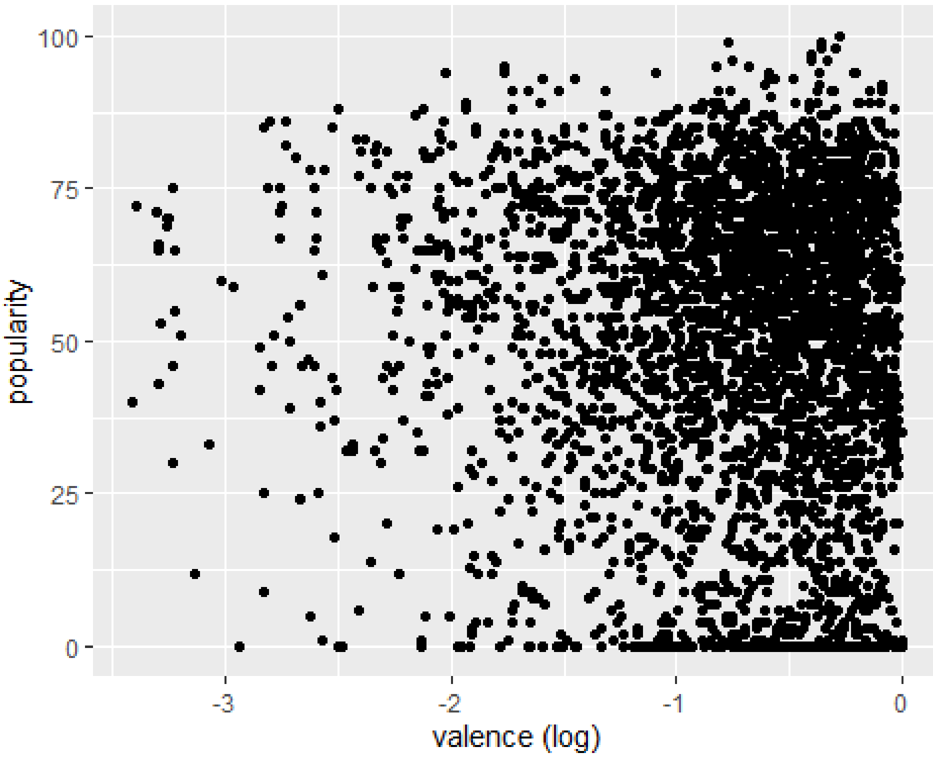

```
# without confidence intervals around smoothed line; no standard error (se)
TikTok.data %>%
  ggplot(aes(x = valence, y = popularity)) +
  geom_point() +
  geom_smooth(method = "",  se = TRUE, color = "red")

## `geom_smooth()` using formula 'y ~ x'

## Warning: Computation failed in `stat_smooth()`:
## invalid first argument
```



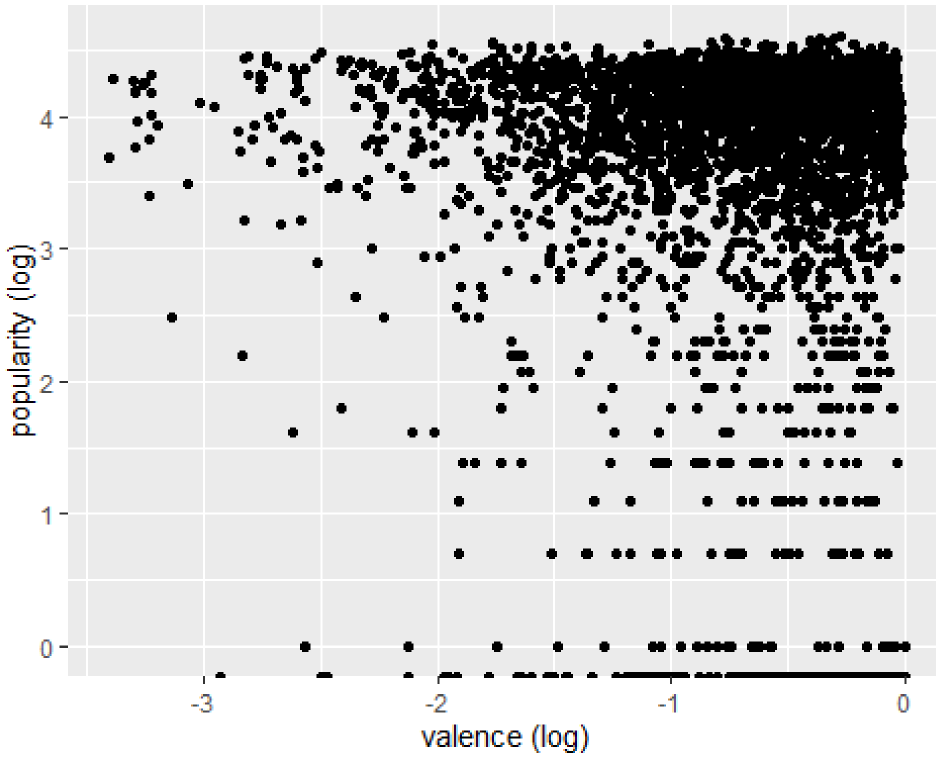

```
###Algorithm chooses the  non-linear model for our data.
###To find a model that is the best model for our data, we transform our vari
able using log log transformation. we use linear log, log lienar, and then lo
g log.


# Transforming variables
#   try to use logarithmic transformation on popularity and/or on valence
#   maybe we can obtain linear relation with transformed variables

# transformation manually (natural logarithm ln=log):
TikTok.data <- TikTok.data %>%
  mutate(`popularity (log)` = log(popularity),
         `valence (log)` = log(valence))

# log popularity VS valence - not linear relation :(
TikTok.data %>%
  ggplot(aes(x = valence, y = `popularity (log)`)) +
  geom_point()
```



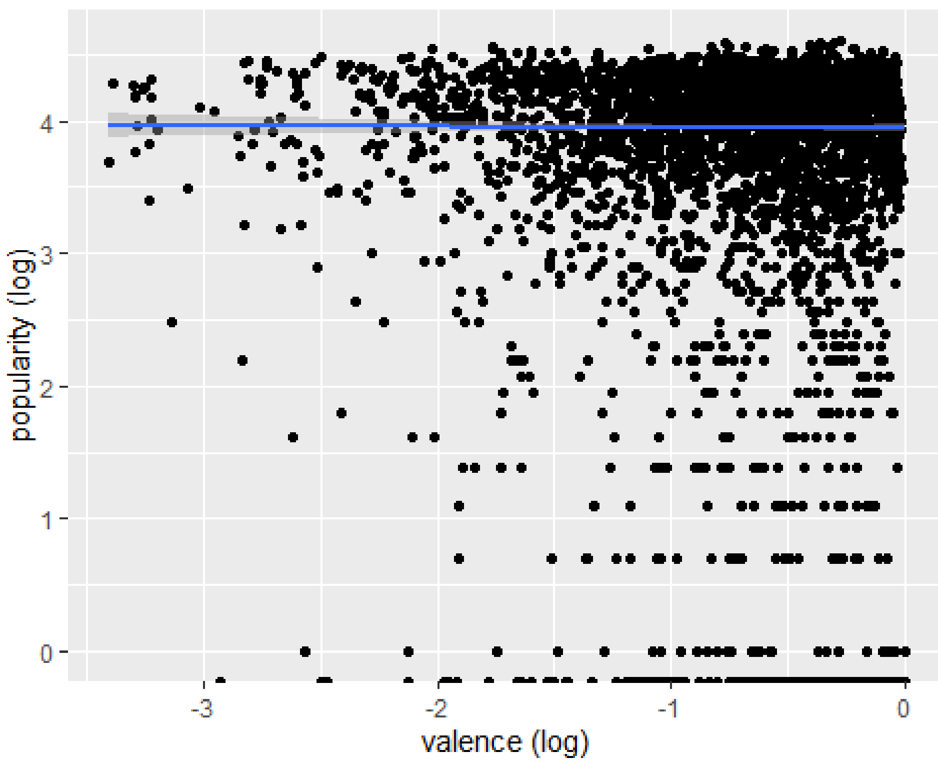

```
# popularity VS log valence - not linear relation :(
TikTok.data %>%
  ggplot(aes(x = `valence (log)`, y = popularity)) +
  geom_point()
```



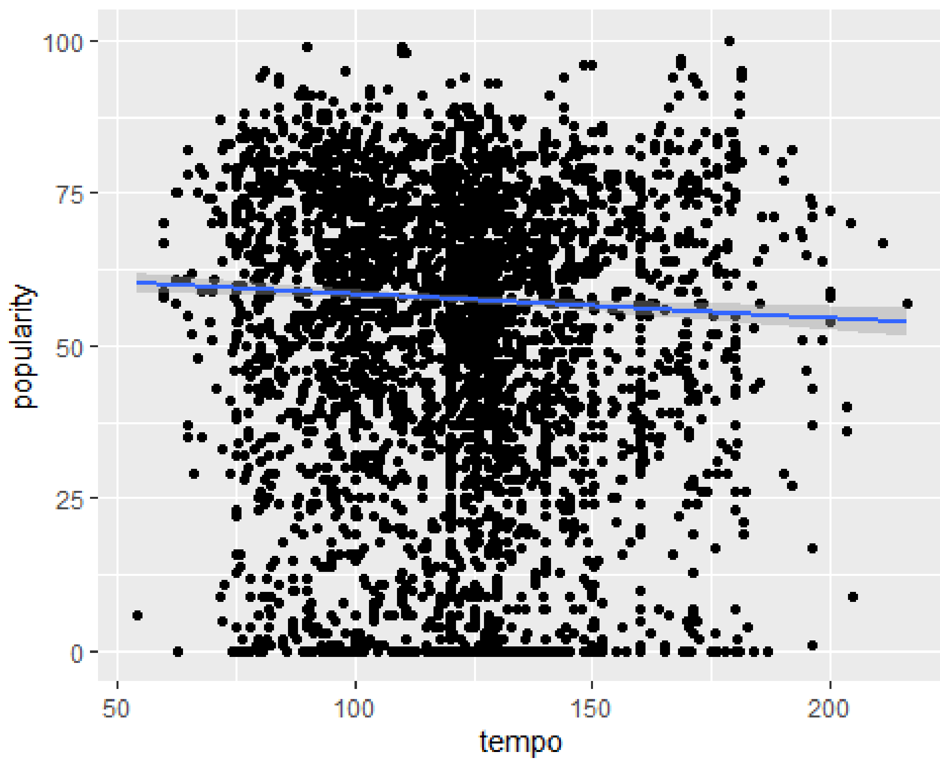

```
# log popularity VS log valence - not linear relation:(
TikTok.data %>%
  ggplot(aes(x = `valence (log)`, y = `popularity (log)`)) +
  geom_point()
```



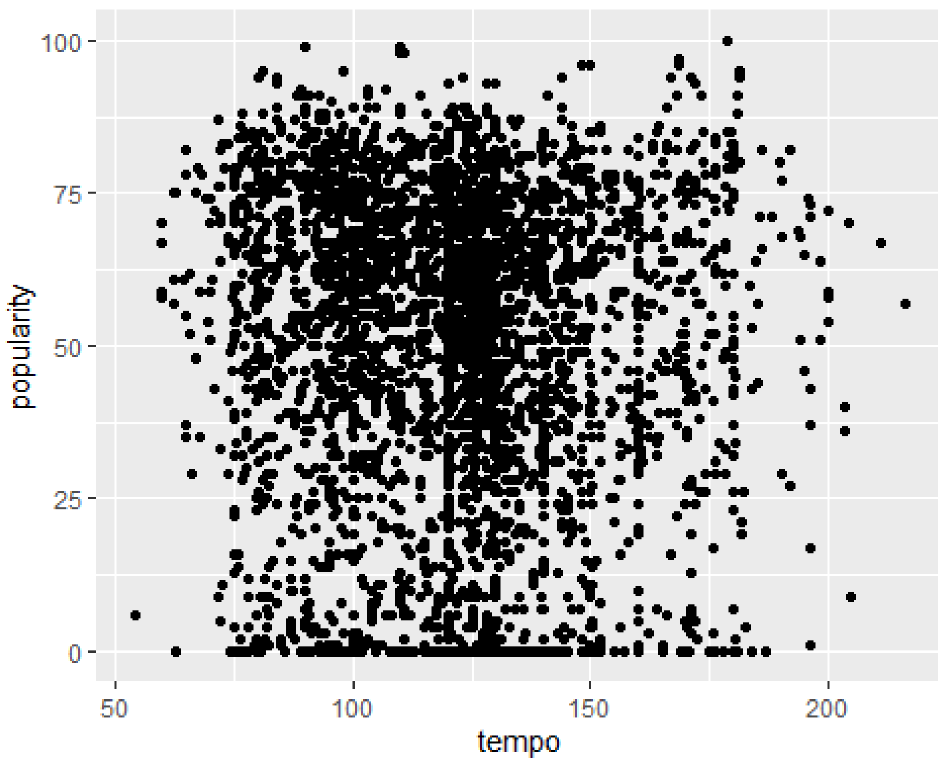

```
# let's fit a model on this transformed variables
TikTok.data %>%
  ggplot(aes(x = `valence (log)`, y = `popularity (log)`)) +
  geom_point() +
  geom_smooth(method = "lm")

## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 276 rows containing non-finite values (stat_smooth).
```



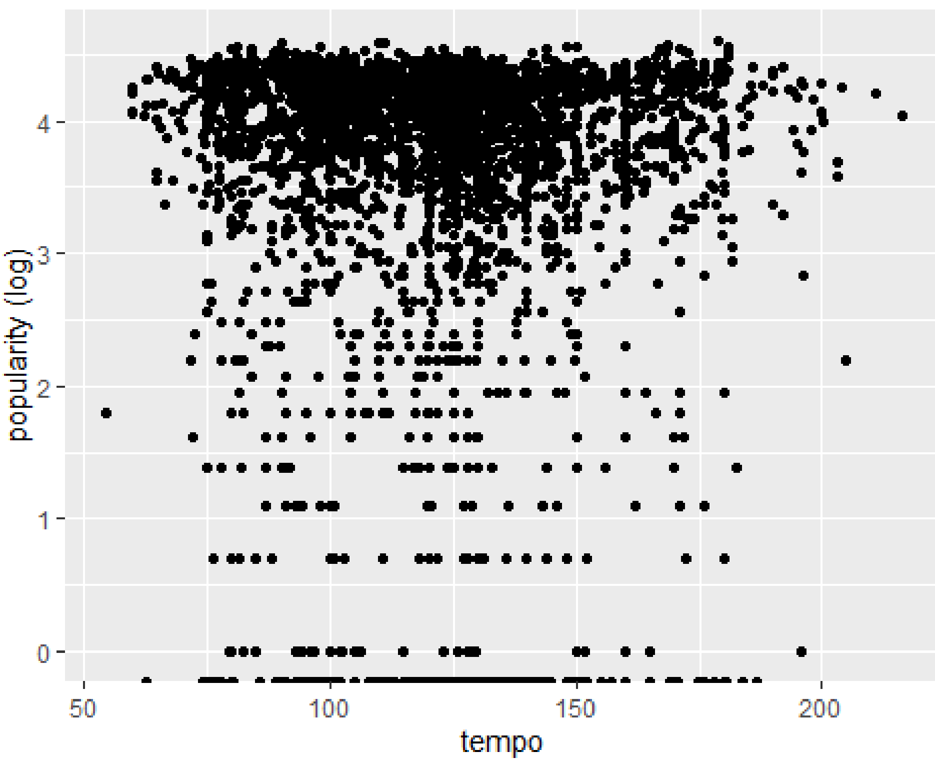

```
###tempo
# use geom_smooth - for adding regression model
TikTok.data %>%
  ggplot(aes(x = tempo, y = popularity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = 'y ~ x')
```



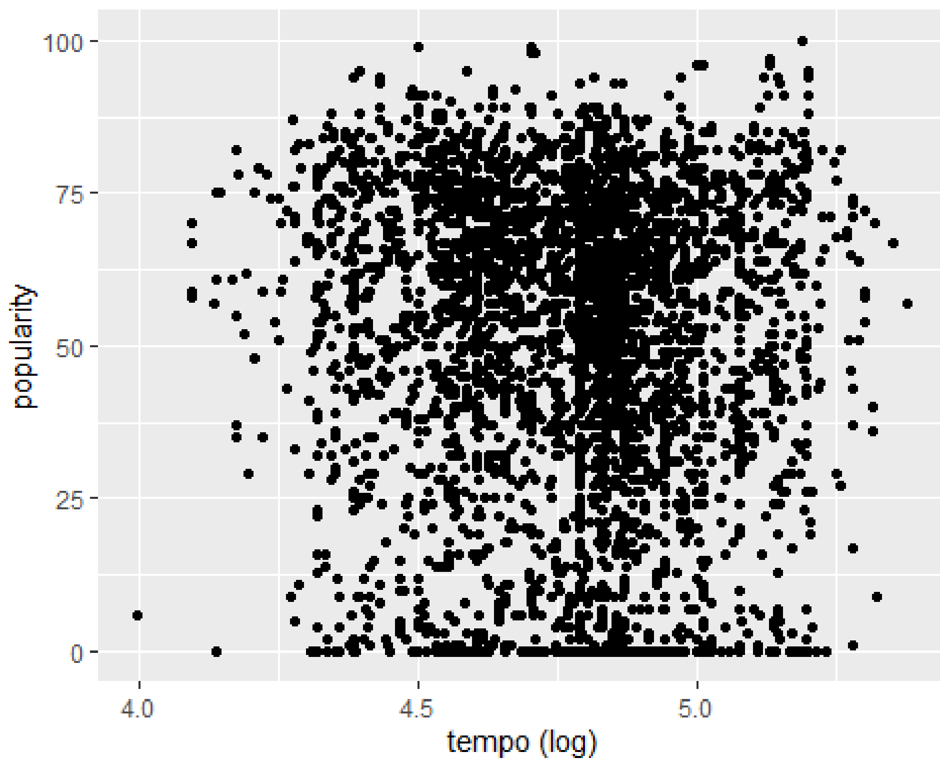

```
# without confidence intervals around smoothed line; no standard error (se)
TikTok.data %>%
  ggplot(aes(x = tempo, y = popularity)) +
  geom_point() +
  geom_smooth(method = "",  se = TRUE, color = "red")

## `geom_smooth()` using formula 'y ~ x'

## Warning: Computation failed in `stat_smooth()`:
## invalid first argument
```



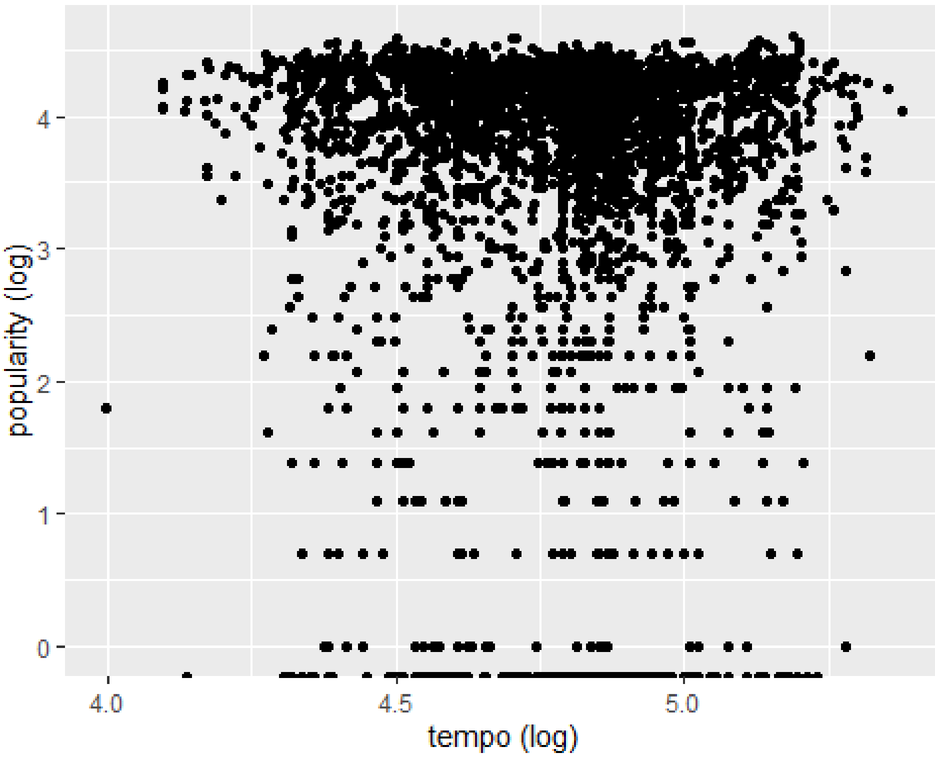

```
###Algorithm chooses the  non-linear model for our data.
###To find a model that is the best model for our data, we transform our vari
able using log log transformation. we use linear log, log lienar, and then lo
g log.


# Transforming variables
#   try to use logarithmic transformation on popularity and/or on tempo
#   maybe we can obtain linear relation with transformed variables

# transformation manually (natural logarithm ln=log):
TikTok.data <- TikTok.data %>%
  mutate(`popularity (log)` = log(popularity),
         `tempo (log)` = log(tempo))

# log popularity VS tempo - not linear relation :(
TikTok.data %>%
  ggplot(aes(x = tempo, y = `popularity (log)`)) +
  geom_point()
```



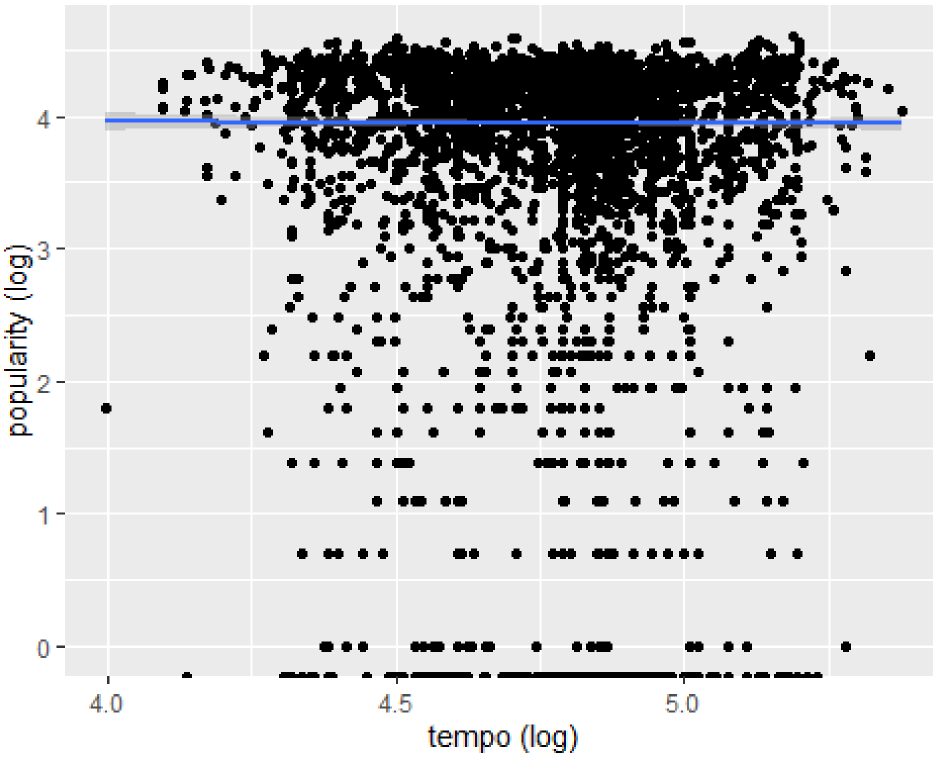

```
# popularity VS log tempo - not linear relation :(
TikTok.data %>%
  ggplot(aes(x = `tempo (log)`, y = popularity)) +
  geom_point()
```




```
# log popularity VS log tempo - not linear relation:(
TikTok.data %>%
  ggplot(aes(x = `tempo (log)`, y = `popularity (log)`)) +
  geom_point()
```



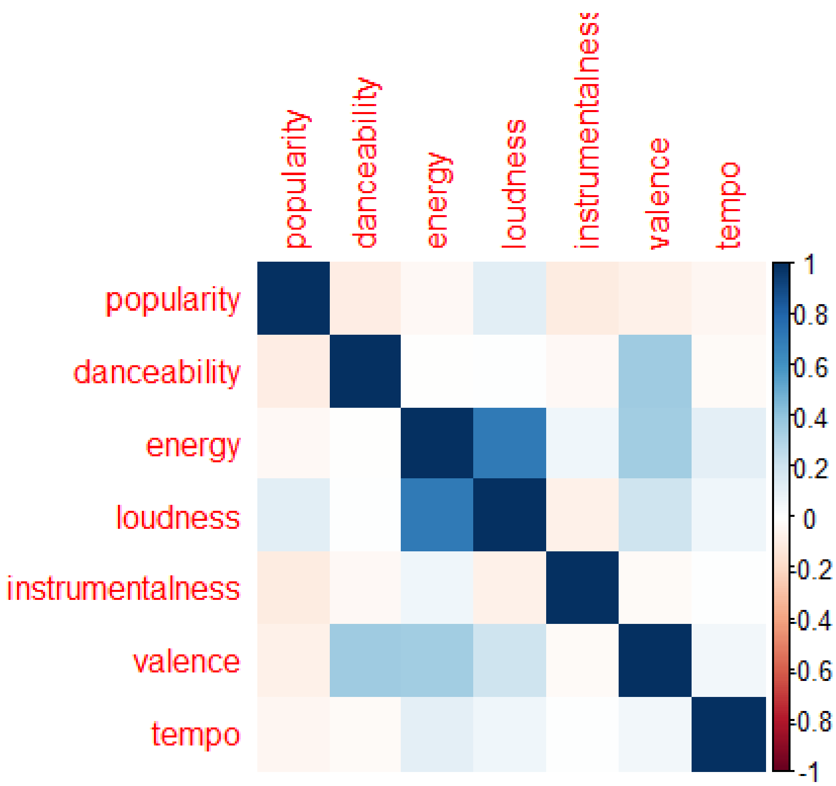

```
# let's fit a model on this transformed variables
TikTok.data %>%
  ggplot(aes(x = `tempo (log)`, y = `popularity (log)`)) +
  geom_point() +
  geom_smooth(method = "lm")

## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 276 rows containing non-finite values (stat_smooth).
```



The Pearson correlation coefficients for all pairwise association are shown in following plot. The results show that there is strong correlation between energy and loudness. Energy is also correlated with with valence. These results suggest that we probably need to drop valence and energy.

```
cor(TikTok.data [,c(7, 8,9, 10,14, 16,17)])

##                   popularity danceability       energy    loudness
## popularity        1.00000000 -0.094163371 -0.033400555  0.12026617
## danceability     -0.09416337  1.000000000 -0.009338387  0.00425881
## energy           -0.03340055 -0.009338387  1.000000000  0.70519130
## loudness          0.12026617  0.004258810  0.705191296  1.00000000
## instrumentalness -0.10598814 -0.038356743  0.068226941 -0.07585762
## valence          -0.07463710  0.359940211  0.341110113  0.20882551
## tempo            -0.04117815 -0.029166685  0.119158121  0.06764341
##                  instrumentalness     valence        tempo
## popularity           -0.105988145 -0.07463710 -0.041178146
## danceability         -0.038356743  0.35994021 -0.029166685
## energy                0.068226941  0.34111011  0.119158121
## loudness             -0.075857622  0.20882551  0.067643407
## instrumentalness      1.000000000 -0.02969654  0.002186916
## valence              -0.029696541  1.00000000  0.054052758
## tempo                 0.002186916  0.05405276  1.000000000

L = cor(TikTok.data [,c(7, 8,9, 10,14, 16,17)])
corrplot(L, method = 'number') # colorful number
```


```
corrplot(L, method = 'color')
```



TikTok popularity prediction models In order to identify the TikTok popularity linear regression model, I will try different predictor variables (different number of predictors and different

predictors). For this purpose, I won't apply any variable transformation(popularity will be used, not log(popularity)). I will use the cross-validation technique, so 80% data - train dataset and 20% data - test dataset are used to identify the possible predictors to predict TikTok tracks' popularity. In this method, I consider bias ~ variance trade off and Root Mean Square Error (RMSE) metrics used to compare differernt models.

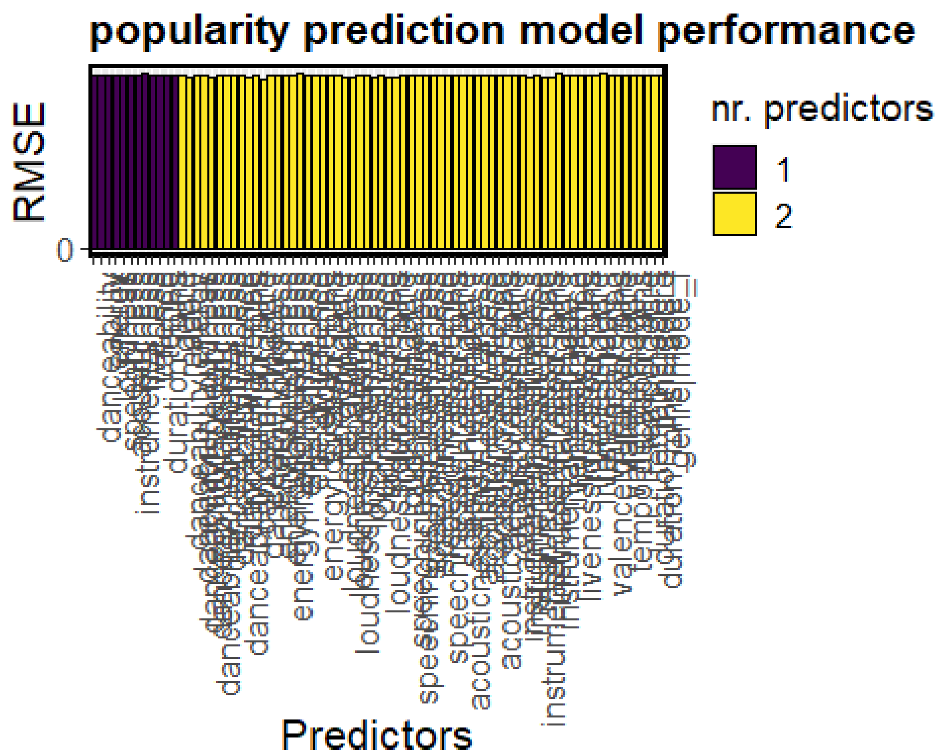
```
# first lets find out how many different models we can build
colnames(TikTok.data)

##  [1] "track_id"          "track_name"        "artist_id"
##  [4] "artist_name"       "album_id"          "release_date"
##  [7] "popularity"        "danceability"      "energy"
## [10] "loudness"          "mode"              "speechiness"
## [13] "acousticness"      "instrumentalness"  "liveness"
## [16] "valence"           "tempo"             "playlist_id"
## [19] "playlist_name"     "duration_mins"     "genre"
## [22] "mode_f"            "popularity (log)"  "danceability (log)"
## [25] "loudness (log)"    "speechiness (log)" "acousticness (log)"
## [28] "liveness (log)"    "valence (log)"     "tempo (log)"

possible.predictors <- colnames(TikTok.data)[c(8,9,10,12, 13, 14, 15, 16,17,
20, 21, 22)] # possible predictors
possible.predictors

##  [1] "danceability"     "energy"           "loudness"         "speechiness
"
##  [5] "acousticness"     "instrumentalness" "liveness"         "valence"
##  [9] "tempo"            "duration_mins"    "genre"            "mode_f"

# generate all possible combinations of predictors
df.models <- NULL # here we store all possible models
model.count <- 1 # counter for model

for(nr.predictors in 1:2){

  predictors <- combn(x = possible.predictors, m = nr.predictors) # generate
all possible combinations of predictors

  for(combination in 1:ncol(predictors)){ # loop over every combination of pr
edictors

    predictors.list <- paste0(predictors[,combination], collapse = "|") # all
predictors
    formula <- paste0("popularity~",paste0(predictors[,combination], collapse
= "+")) # model formula

    df.models <- rbind(df.models, c(model.count, nr.predictors, predictors.li
st, formula))
    model.count <- model.count + 1 # increase model count
  }
```

```r
}

colnames(df.models) <- c("id", "nr. predictors", "predictors", "formula") # c
olumn names

#convert data in data frame and cread a numeric id column.
df.models <- df.models %>%
  as.data.frame() %>% # convert to data frame
  mutate(id = as.numeric(as.character(id))) # convert to numeric

# Generate all possible models (with function)
#TikTok.models <- generate.models(predictors.vars = possible.predictors, outc
ome.var = "popularity")

# Split data frame train ~ test
set.seed(123)
id.rows <- 1:nrow(TikTok.data) # all rows ids
id.train <- sample(x = id.rows, size = round(0.8 * nrow(TikTok.data)), replac
e = F) # train rows
id.test <- setdiff(id.rows, id.train) # test rows- removes the ro


# # write back to data frame; add a new col name sample
TikTok.data[id.train,"sample"] <- "train"
TikTok.data[id.test,"sample"] <- "test"
#check the test and train size
TikTok.data%>% count(sample)
```

```
##   sample    n
## 1   test 1349
## 2  train 5397
```

```r
# Split sample (with function)
set.seed(123)

#TikTok.data <- split.sample(TikTok.data)
TikTok.data.train <- TikTok.data %>% filter(sample == "train")
TikTok.data.test <- TikTok.data %>% filter(sample == "test")


# Train models (train dataset), predict popularity (test dataset), calculate
RMSE (test dataset)

df.models <- df.models %>%# add RMSE column
  mutate(RMSE = NA)

for(id.model in 1:nrow(df.models)){ # loop over each model
  # train model
  formula <- df.models[id.model,"formula"]
```

```
  lm.model <- lm(formula = formula, data = TikTok.data.train)

  # predict popularit (test dataset)
  TikTok.data.test <- TikTok.data.test %>%
    mutate(`popularity predicted` = predict(lm.model, .))

  # calculate RMSE (predicted popularity VS actual popularity)
  SSE <- (TikTok.data.test$`popularity predicted` - TikTok.data.test$populari
ty)^2 # sum of squared errors
  RMSE <- sqrt(mean(SSE))
  # write RMSE back to table
  df.models[id.model,"RMSE"] <- RMSE
}
#Small RMSE indicates a better prediction
# draw model performance
predictors.levels <- df.models %>% arrange(id) %>% pull(predictors) # levels
for predictors factor variable


df.models %>%
  mutate(`nr. predictors` = as.factor(as.character(`nr. predictors`)),
         predictors = factor(predictors, levels = predictors.levels)) %>%
  ggplot(aes(x = predictors, y = RMSE, fill = `nr. predictors`)) +
  geom_bar(stat = "identity", color = "black") +
  scale_y_continuous(breaks = seq(0, 5000, 500)) +
  scale_fill_viridis_d() +
  xlab("Predictors") +
  ylab("RMSE") +
  ggtitle("popularity prediction model performance") +
  theme(axis.text = element_text(size = 12),
        axis.text.x = element_text(angle = 90, hjust = 1),
        axis.title = element_text(size = 16),
        plot.title = element_text(size = 16, face = "bold"),
        legend.title = element_text(size = 14),
        legend.text = element_text(size = 12),
        panel.border = element_rect(color = "black", fill = NA, size = 1.5))
```



According to the cross-validation technique, varibles danceablity, loudness, instrumentalness, tempo, and genre are the best variables to predict the popularity of TikTok tracks. As such, we fit the reduced model.

```
# Fit the reduced model

md2 <- lm(popularity~danceability + loudness + instrumentalness + tempo)
summary(md2)
```

```
##
## Call:
## lm(formula = popularity ~ danceability + loudness + instrumentalness +
##     tempo)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -68.092 -12.637   6.146  17.926  45.360
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       84.54290    2.30914  36.612  < 2e-16 ***
## danceability     -17.85122    2.13612  -8.357  < 2e-16 ***
## loudness           1.00795    0.10395   9.696  < 2e-16 ***
## instrumentalness -17.95099    2.13558  -8.406  < 2e-16 ***
## tempo             -0.04978    0.01153  -4.317 1.61e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

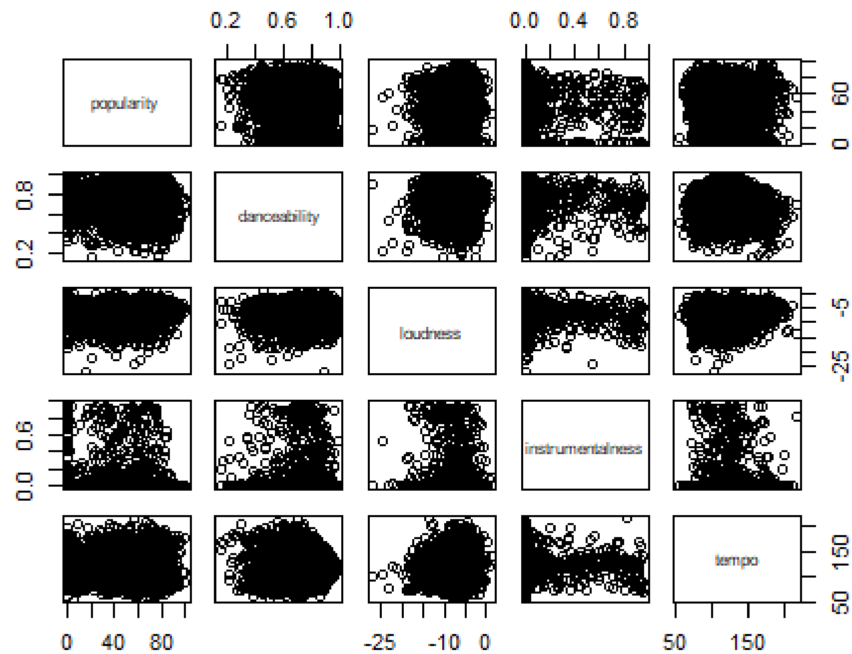
```
##
## Residual standard error: 24.17 on 6741 degrees of freedom
## Multiple R-squared:  0.03626,    Adjusted R-squared:  0.03569
## F-statistic:  63.4 on 4 and 6741 DF,  p-value: < 2.2e-16
```

A. Diagnostics for Predictors.

The purpose of this section is to examine the distribution of predictors, identify any unusually large or small values, and examine bivariate associations to identify multicollinearity. Unusual values should be flagged as they may **influence** the fit of the model. Bivariate associations between predictors could cause issues if the purpose of the model is estimation.

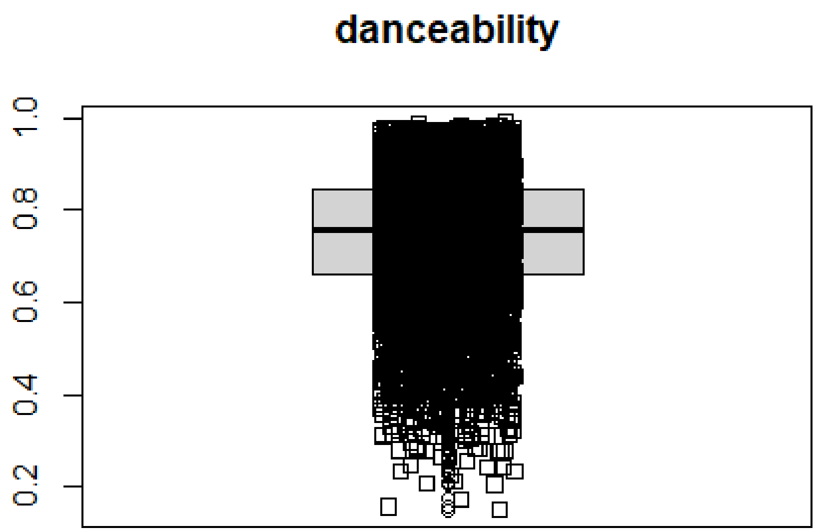
tThe following scatterplot matrix indicates the associations between all variables.

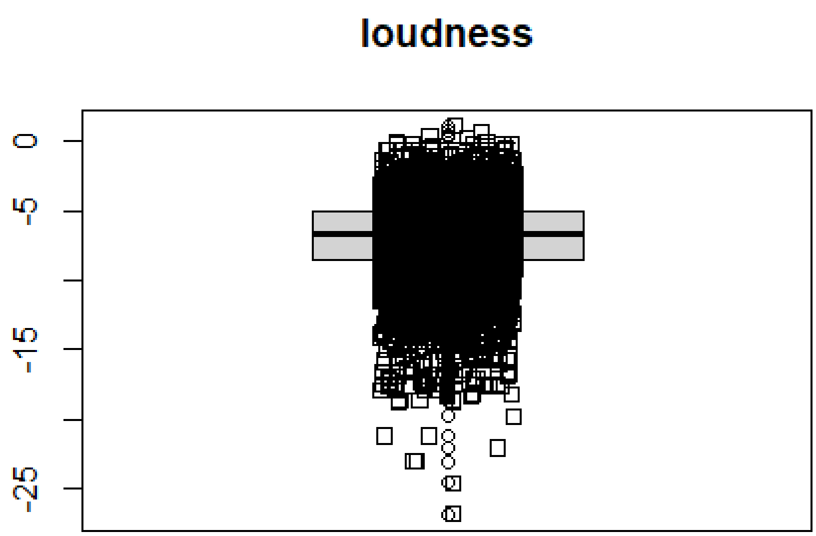
```
pairs(popularity~danceability + loudness + instrumentalness + tempo)
```

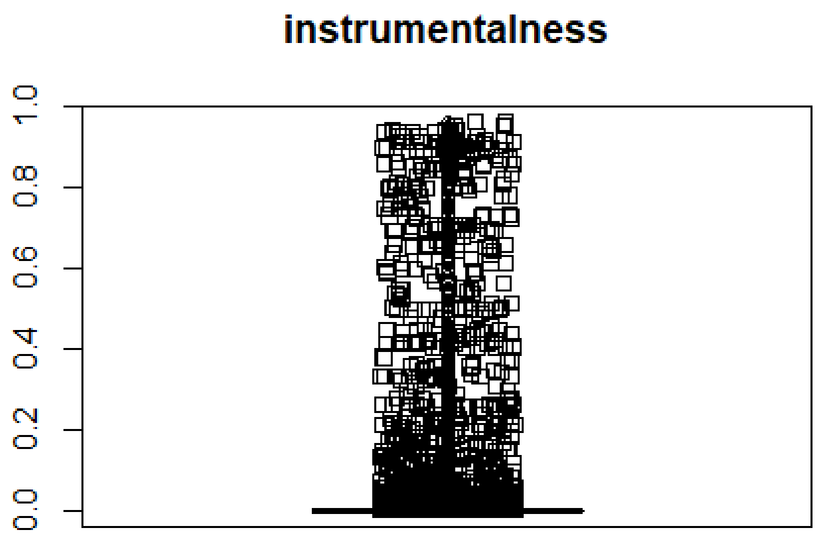


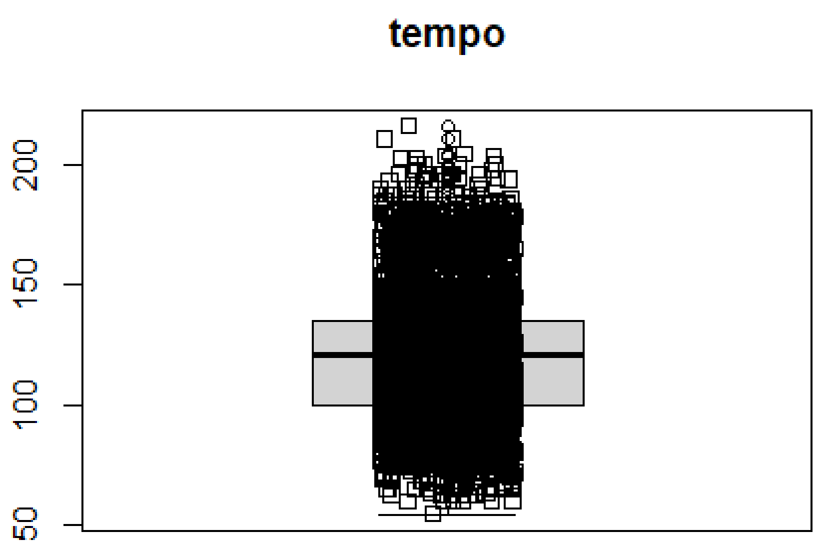
Strip plots for all predictors and the dependent variable (jittered) are shown next to boxplots of the same data.

```
for (i in c(8,10,14, 17)){
  boxplot(TikTok.data[,i], main = names(TikTok.data)[i])
  stripchart(TikTok.data[,i], vertical = T, method = "jitter", add = TRUE)
}
```


danceability
1.0
0.8
0.6
0.4
0.2


loudness
0
-5
-15
-25



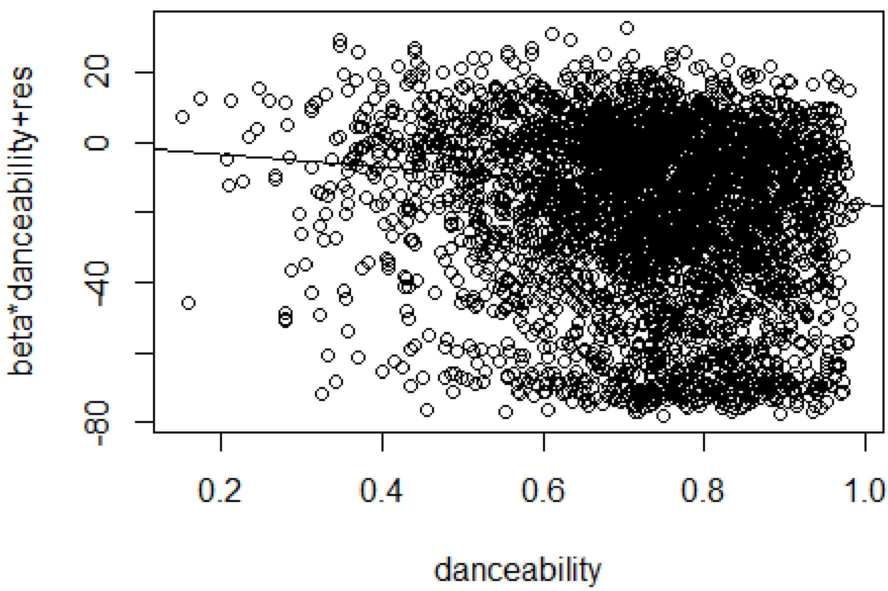


C. Screening of Predictors

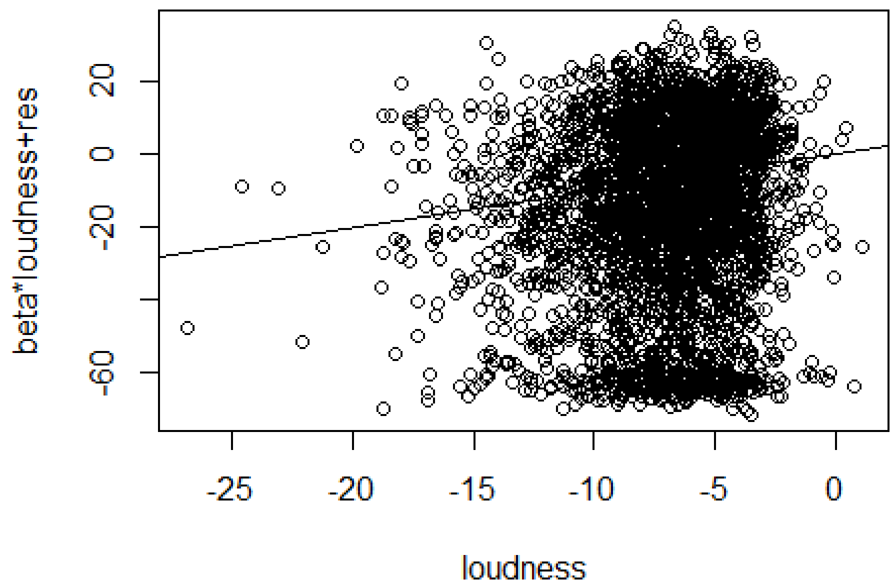
1. **Added variable plots** for each of the covariates are shown. Added variable plots (also known as partial residual plots or adjusted variable plots) provide evidence of

the importance of a covariate given the other covariates already in the model. They also display the nature of the relationship between the covariate and the outcome (i.e., linear, curvilinear, transformation necessary, etc.) and any problematic data points with respect to the predictor. The plots all indicate no need for transformations because linear relationships are apparent. They also indicate each variable provides some added value to a model that already includes all other covariates because the slopes of the linear relationships are all appear to be non-zero.

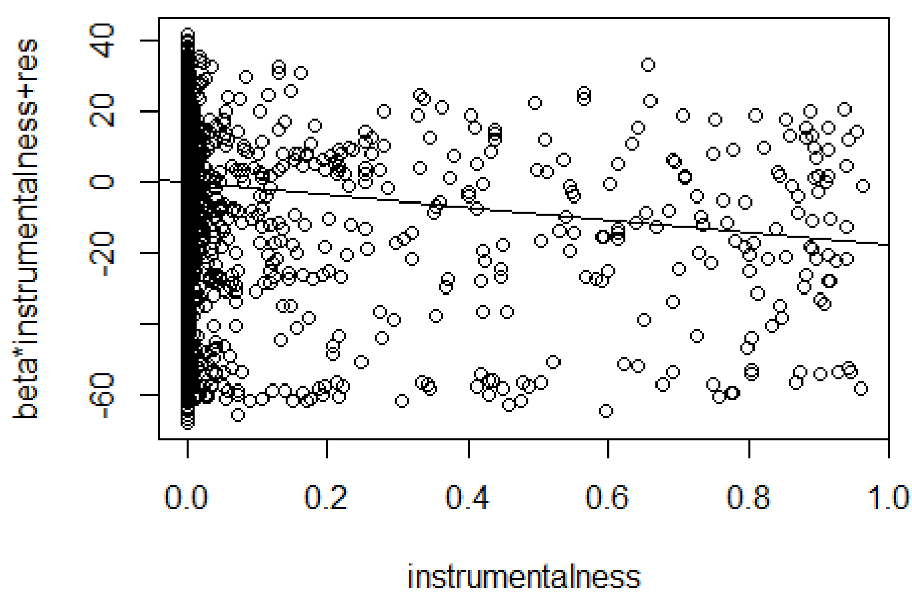
```
prplot(md2,1)
```



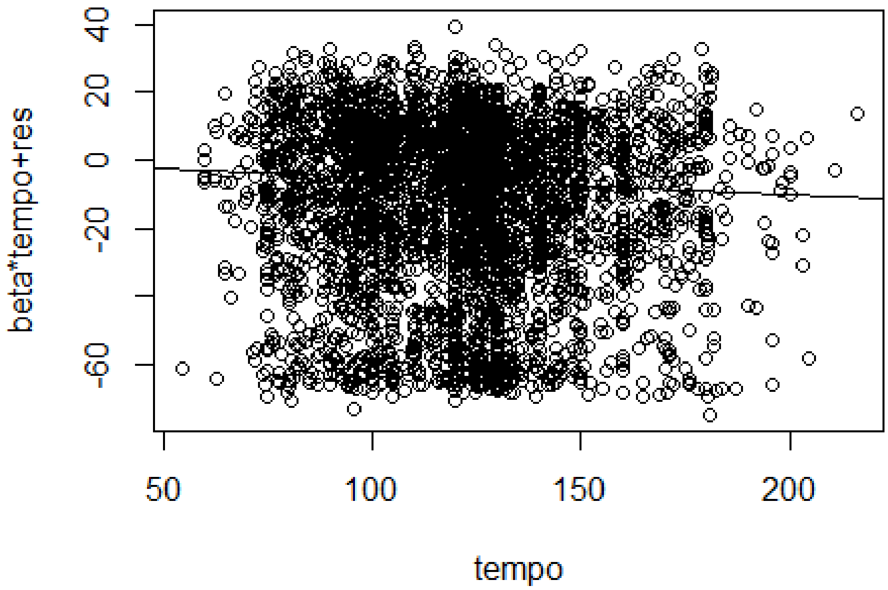

```
prplot(md2,2)
```




```
prplot(md2,3)
```




```
prplot(md2,4)
```



2. Since the purpose of the project is to identify variables that predict the popularity of TikTok tracks, the goal is estimating coefficients. **Multicollinearity** can create instability in estimation and so it should be avoided. **Variance inflation factors (VIF)** measure how much the variances of the estimated regression coefficients are inflated as compared to when the predictor variables are not linearly related. A maximum VIF in excess of 10 is a good rule of thumb for multicollinearity problems. Based on the maximum VIF, for the reduced model, there do not appear to be any issues that need remediation. However, I received the error "there are aliased coefficients in the model" for the initial fitted model which refers to the existence of perfect multicollinearity.

```
vif(md2)

##     danceability         loudness instrumentalness            tempo
##         1.002334         1.010472         1.007292         1.005512
```

3. **matic variable selection methods** can be a useful starting point in eliminating redundant variables. They should only be used as a guide to the screening and removal (or addition) of predictors.

```
fa <- regsubsets(popularity~danceability + instrumentalness + tempo + loudnes
s + energy + valence, data=TikTok.data)
fma <- summary(fa)
fma

## Subset selection object
## Call: regsubsets.formula(popularity ~ danceability + instrumentalness +
##     tempo + loudness + energy + valence, data = TikTok.data)
```

```
## 6 Variables  (and intercept)
##                    Forced in Forced out
## danceability           FALSE      FALSE
## instrumentalness       FALSE      FALSE
## tempo                  FALSE      FALSE
## loudness               FALSE      FALSE
## energy                 FALSE      FALSE
## valence                FALSE      FALSE
## 1 subsets of each size up to 6
## Selection Algorithm: exhaustive
##          danceability instrumentalness tempo loudness energy valence
## 1  ( 1 ) " "          " "              " "   "*"      " "    " "
## 2  ( 1 ) " "          " "              " "   "*"      "*"    " "
## 3  ( 1 ) "*"          " "              " "   "*"      "*"    " "
## 4  ( 1 ) "*"          "*"              " "   "*"      "*"    " "
## 5  ( 1 ) "*"          "*"              "*"   "*"      "*"    " "
## 6  ( 1 ) "*"          "*"              "*"   "*"      "*"    "*"
```

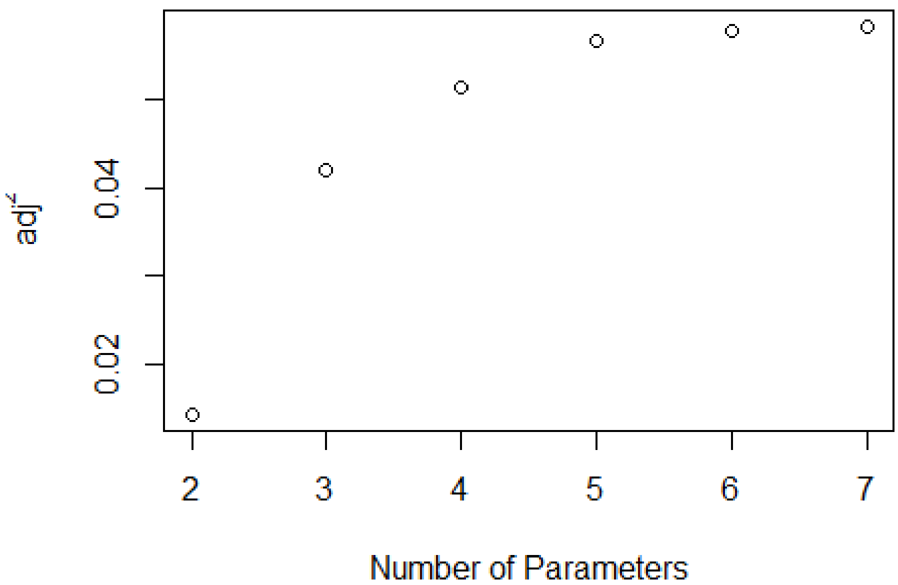
The summary output includes a matrix indicating which predictors are included in each of the 6 candidate models. In the first model (first row of the matrix, only loudness is included. In the third model (row 3) with three (indicated by a '3') predictors, danceablity, loudness, and energy are included. The fourth, fifth, and sixth models are in the last rows of the matrix.

Several criteria for selecting the best model are produced, including R^2_{adj} (large values are better), Bayes Information Criterion BIC (smaller values are better), Bayes Information Criterion BIC (smaller values are better), and Mallow's C_p statistic (values of C_p close to p (number of beta coefficients). Other criteria not produced by the `regsubsets` function are AIC and $PRESS$. I calculate these statistics for the two potential final models based on the results of automatic variable selection. As we have B0, we add one to the number of parameters, so instead of 1:6, we have 2:7. Here, all statistics indicate that the best model is one in which energy and valence are removed. The second best is the full model.

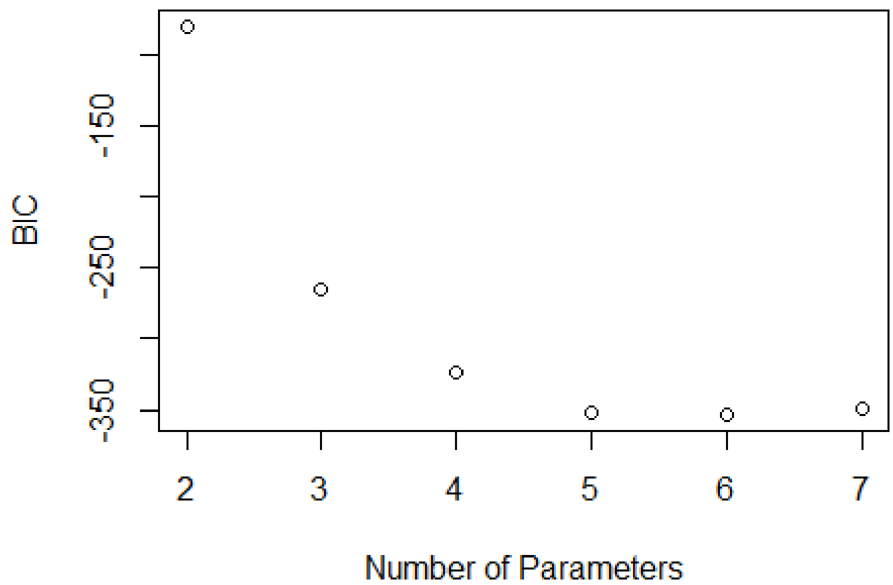
```r
fma$adj # Adjusted R2 bigger better
```

```
## [1] 0.01431782 0.04197727 0.05136388 0.05655621 0.05773590 0.05821183
```

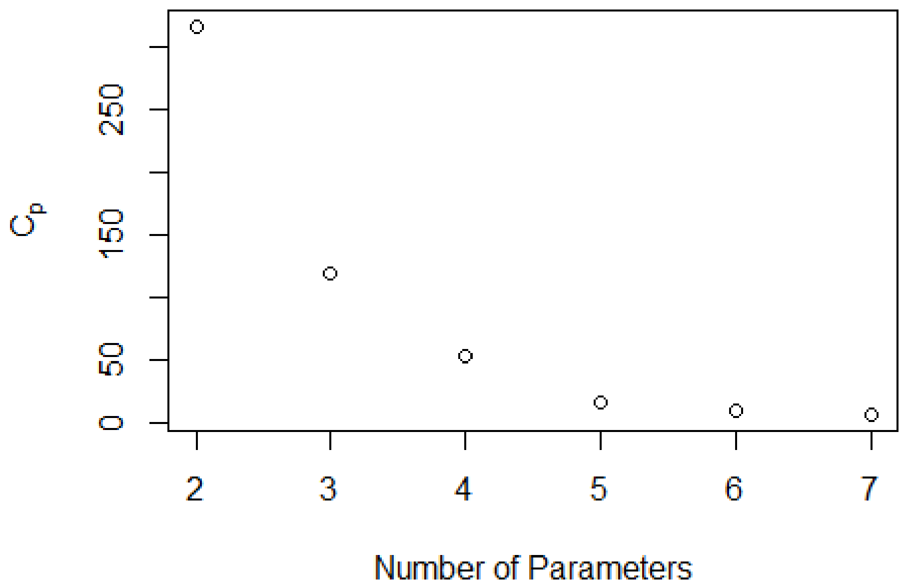
```r
plot(2:7,fma$adj, xlab = "Number of Parameters", ylab = expression(adj^2))
```




```
##The increase in the last two variables is not significant.
fma$bic # BIC small

## [1]  -80.65294 -264.84441 -323.45077 -352.66020 -353.28483 -348.87732

plot(2:7, fma$bic, xlab = "Number of Parameters", ylab = expression(BIC))
```




```
##The decrease in the last two variables is not significant.

fma$cp # Cp = p

## [1] 316.318273 119.235968  53.022611  16.850376   9.406064   7.000000

plot(2:7, fma$cp, xlab = "Number of Parameters", ylab = expression(C[p]))
```




```
# Extract AIC
extractAIC(md1)

## [1]     7.00 42822.93

extractAIC(md2)

## [1]     5.00 42980.38

# Extract PRESS
PRESS(md1)

## [1] 3853996

PRESS(md2)

## [1] 3945005
```

C. Model Validation

Model validation can help us select the model that has the best predictive performance in a hold-out sample. There are several approaches to model validation, two of which are shown here.

Leave-one-out cross validation is useful for smaller datasets where training and testing data are not feasible. This method involves:

1. Leave out one data point and build the model using the remaining data.

2. Test the model against the data point removed in Step 1 and record the prediction error.
3. Repeat for all data points.
4. Compute the overall prediction error by averaging the prediction errors.
5. If comparing models, the model with lowest MSPE should be selected.

The MSPE is smaller for the model without energy and valence.

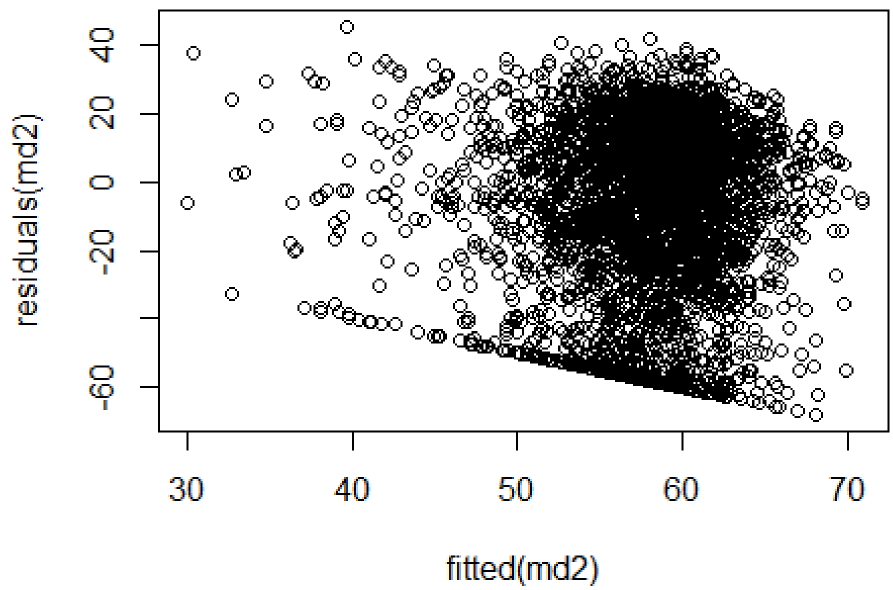
```
# Define the training method
trc <- trainControl(method="LOOCV")

# Train the model
moreduced <- train(popularity~danceability + loudness + instrumentalness + te
mpo, data = TikTok.data, method = "lm", trControl = trc)
print(moreduced)

## Linear Regression
##
## 6746 samples
##    4 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 6745, 6745, 6745, 6745, 6745, 6745, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   24.18247  0.0348275  19.32242
##
## Tuning parameter 'intercept' was held constant at a value of TRUE

mfull <-train(popularity~danceability + loudness + instrumentalness + tempo +
energy + valence, data = TikTok.data, method = "lm", trControl = trc)
print(mfull)

## Linear Regression
##
## 6746 samples
##    6 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 6745, 6745, 6745, 6745, 6745, 6745, ...
## Resampling results:
##
##   RMSE     Rsquared    MAE
##   23.9019  0.05709586  18.96022
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

D. Residual Diagnostics

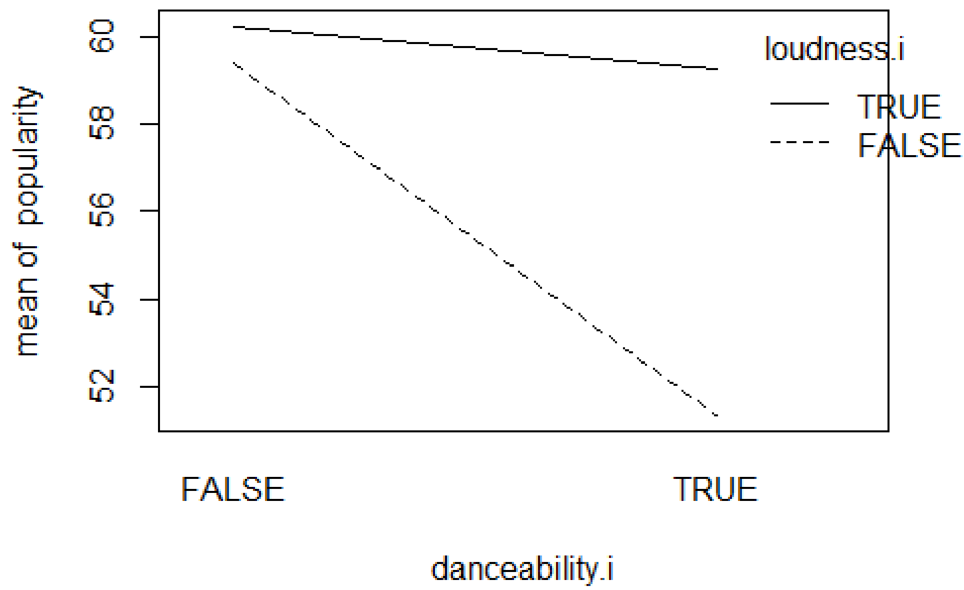
1. Model Completeness

It's a good idea to also check for possible interactions (though we wouldn't hypothesize any for this analysis). The fitted-versus-residual plot looks like noise. This feature results from the nonlinear association.

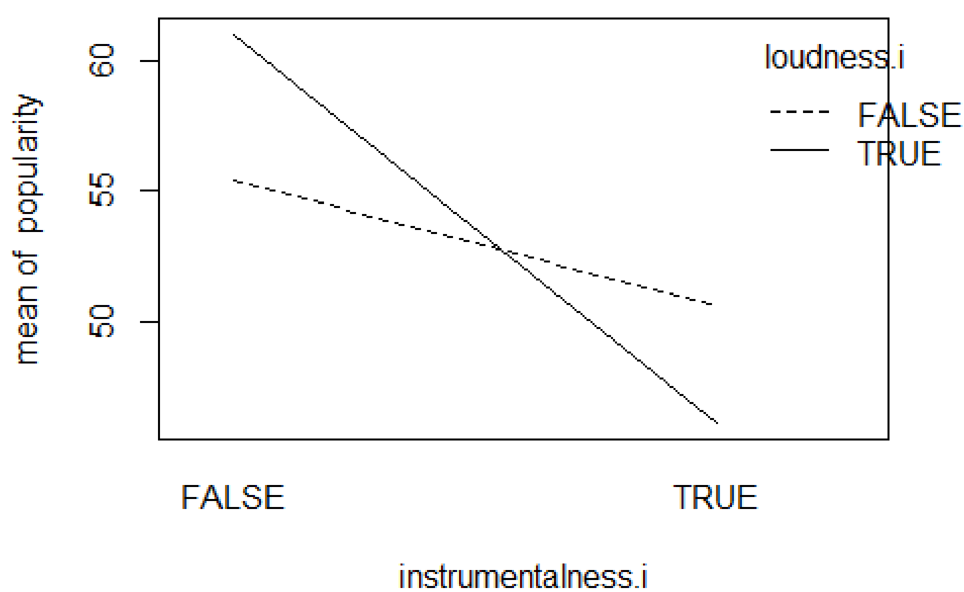
```
plot(residuals(md2)~fitted(md2)) # Model looks appropriate
```



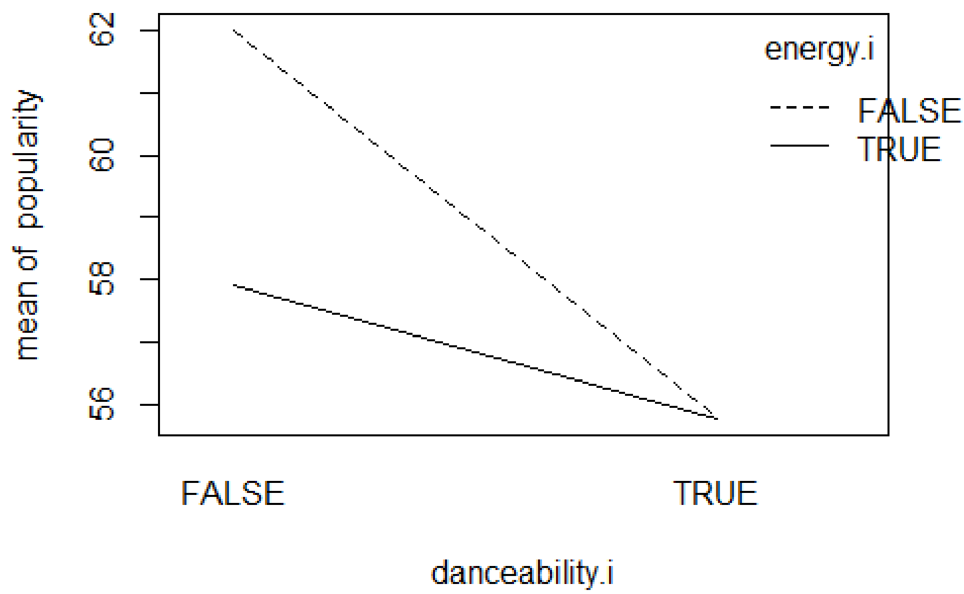

```
danceability.i <- danceability > mean(danceability)
loudness.i <- loudness > mean(loudness)
instrumentalness.i <- instrumentalness > mean(instrumentalness)
tempo.i <- tempo > mean(tempo)
valence.i <- (valence) > mean(valence)
energy.i <- (energy) > mean(energy)

interaction.plot(danceability.i,loudness.i,popularity)
```



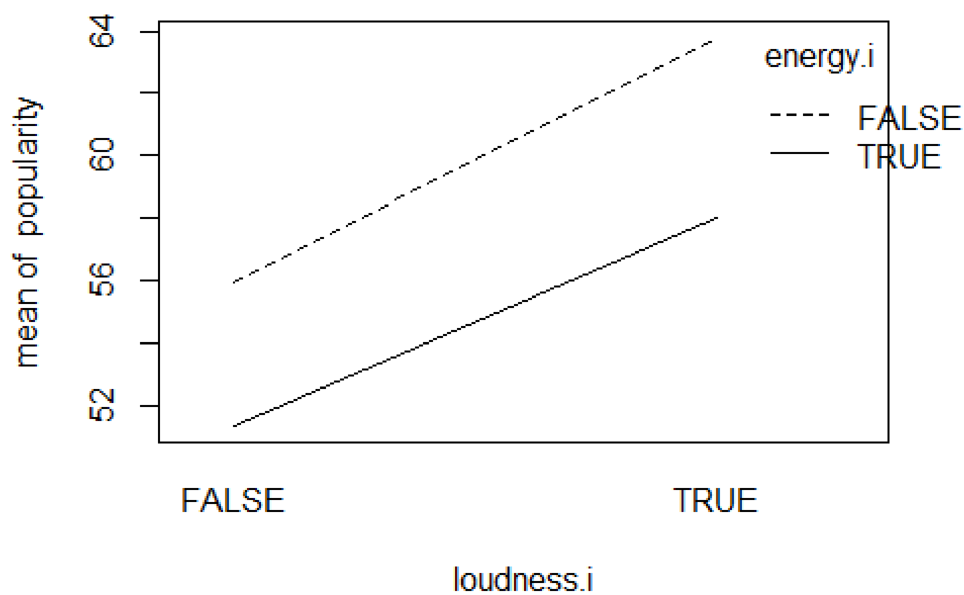

```
interaction.plot(instrumentalness.i,loudness.i,popularity) # Look at that! Is it significant.
```



```
interaction.plot(danceability.i,energy.i,popularity)
```




```
interaction.plot(loudness.i,energy.i,popularity)
```



```r
# Test for significant interaction using general linear f-test
md3 <- lm(popularity~danceability + loudness + instrumentalness + tempo+ inst
rumentalness*loudness)
anova(md3)
```

```
## Analysis of Variance Table
##
## Response: popularity
##                             Df  Sum Sq Mean Sq  F value    Pr(>F)
## danceability                 1   36241   36241  62.0465 3.884e-15 ***
## loudness                     1   59514   59514 101.8918 < 2.2e-16 ***
## instrumentalness             1   41554   41554  71.1426 < 2.2e-16 ***
## tempo                        1   10890   10890  18.6436 1.598e-05 ***
## loudness:instrumentalness    1    2301    2301   3.9397    0.0472 *
## Residuals                 6740 3936794     584
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

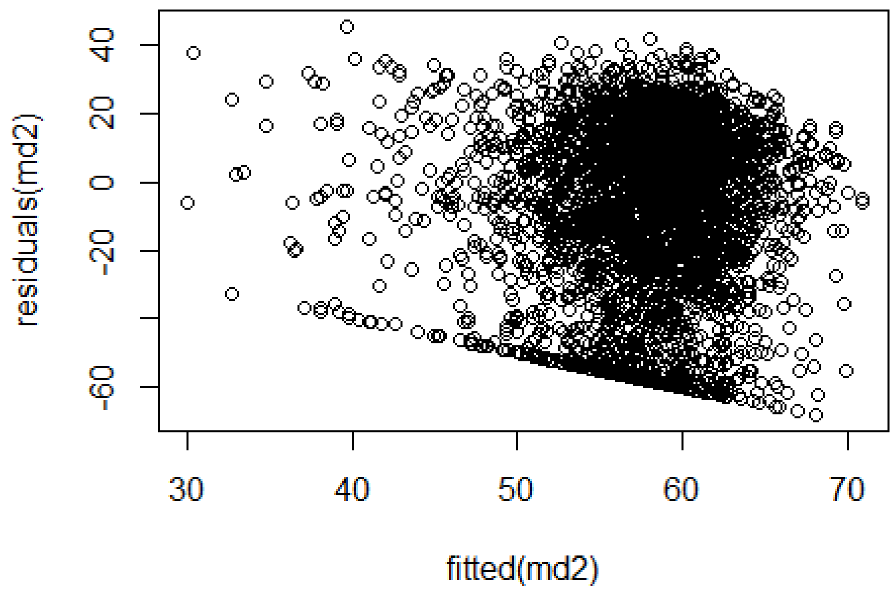
```r
summary(md3)
```

```
##
## Call:
## lm(formula = popularity ~ danceability + loudness + instrumentalness +
##     tempo + instrumentalness * loudness)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -68.252 -12.744   6.184  18.069  41.963
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                84.78353    2.31182  36.674  < 2e-16 ***
## danceability              -17.57259    2.14026  -8.210 2.62e-16 ***
## loudness                    1.07573    0.10940   9.833  < 2e-16 ***
## instrumentalness          -26.49165    4.80349  -5.515 3.62e-08 ***
## tempo                      -0.04943    0.01153  -4.288 1.83e-05 ***
## loudness:instrumentalness  -1.03172    0.51979  -1.985   0.0472 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.17 on 6740 degrees of freedom
## Multiple R-squared:  0.03682,    Adjusted R-squared:  0.03611
## F-statistic: 51.53 on 5 and 6740 DF,  p-value: < 2.2e-16
```

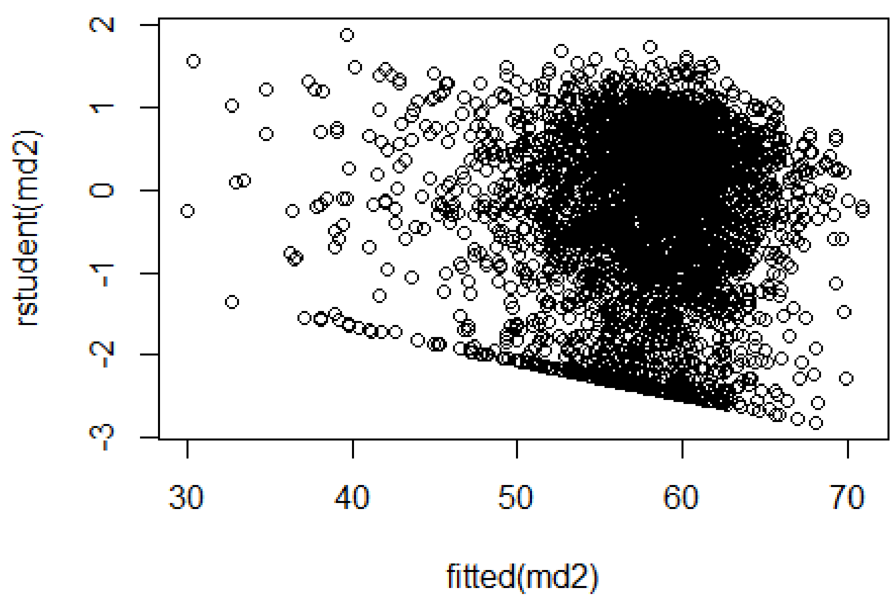
2. *Outliers*

Look for outliers in X and in Y, and also investigate whether there are any influential points.

```r
plot(residuals(md2)~fitted(md2))
```

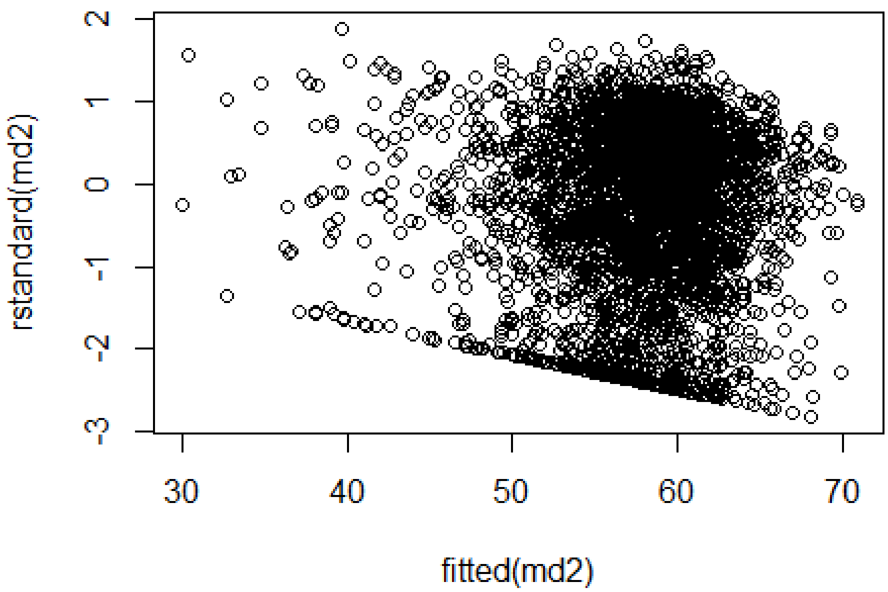



```
plot(rstudent(md2)~fitted(md2)) #Studentized residual (between -3 and 3)
identify(rstudent(md2)~fitted(md2))
```



```
## integer(0)
```

```
plot(rstandard(md2)~fitted(md2)) #Deleted studentized residual
```




```
which(abs(rstandard(md2)) > 3) # No unusual residuals
```

```
## named integer(0)
```

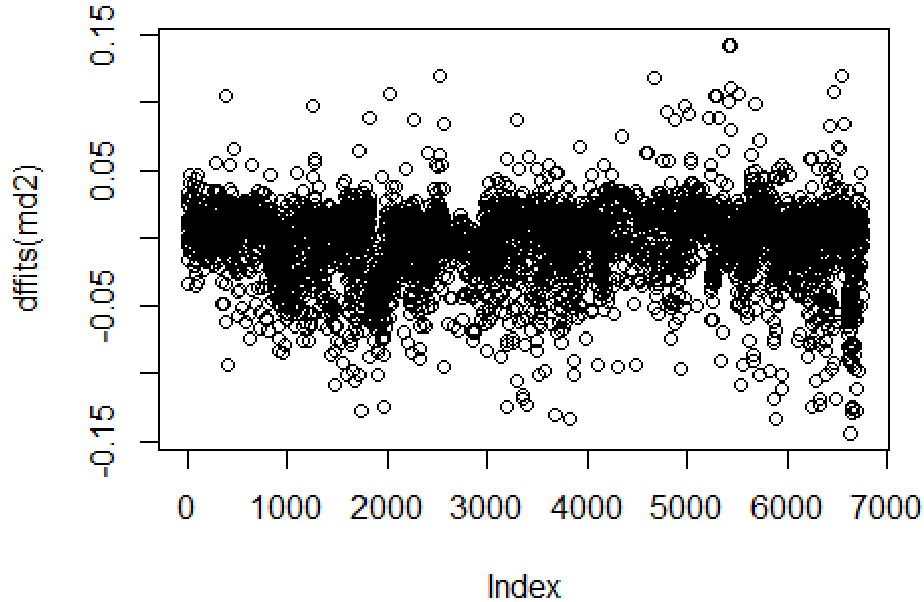
```
which(hatvalues(md2)>2*5/6746) # High leverage?
```

```
##   36   84   85   90   99  100  102  103  104  105  106  108  109  183  188 
189 
##   36   84   85   90   99  100  102  103  104  105  106  108  109  183  188 
189 
##  190  201  222  223  228  304  305  306  320  321  328  329  345  368  385 
400 
##  190  201  222  223  228  304  305  306  320  321  328  329  345  368  385 
400 
##  401  412  413  427  429  477  478  497  499  511  648  659  660  661  662 
663 
##  401  412  413  427  429  477  478  497  499  511  648  659  660  661  662 
663 
##  667  702  740  766  767  768  853  855  908  961 1080 1084 1098 1099 1159 
1204 
##  667  702  740  766  767  768  853  855  908  961 1080 1084 1098 1099 1159 
1204 
## 1205 1206 1207 1211 1264 1268 1273 1279 1280 1315 1327 1331 1349 1353 1450 
1478 
```

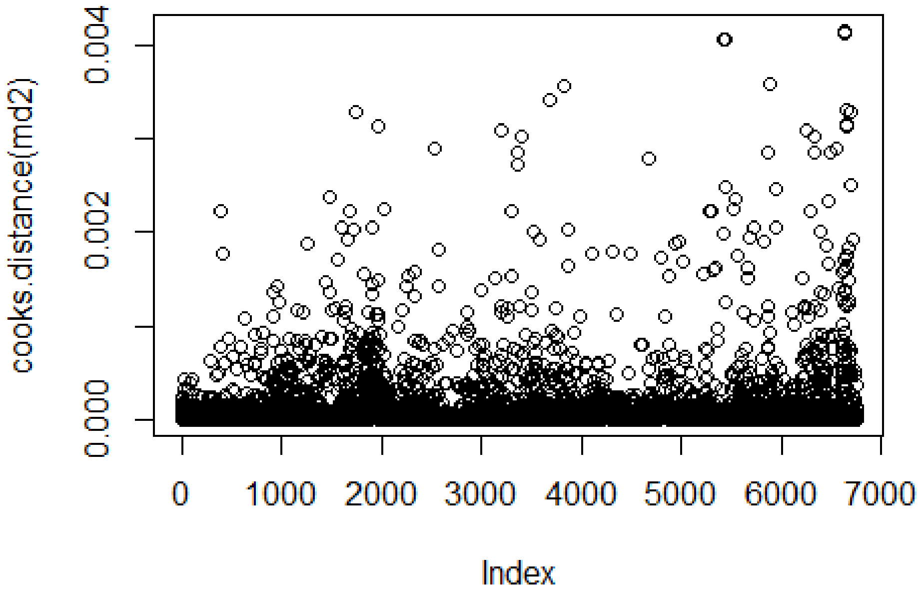
```
## 1205 1206 1207 1211 1264 1268 1273 1279 1280 1315 1327 1331 1349 1353 1450
1478
## 1480 1485 1504 1505 1523 1550 1570 1572 1574 1575 1576 1578 1584 1596 1612
1616
## 1480 1485 1504 1505 1523 1550 1570 1572 1574 1575 1576 1578 1584 1596 1612
1616
## 1617 1641 1651 1654 1665 1678 1681 1704 1717 1721 1725 1743 1745 1771 1772
1797
## 1617 1641 1651 1654 1665 1678 1681 1704 1717 1721 1725 1743 1745 1771 1772
1797
## 1799 1800 1801 1804 1805 1820 1825 1911 1951 1963 1972 1976 1991 2003 2004
2025
## 1799 1800 1801 1804 1805 1820 1825 1911 1951 1963 1972 1976 1991 2003 2004
2025
## 2038 2050 2068 2074 2086 2095 2116 2117 2155 2174 2196 2210 2232 2235 2258
2268
## 2038 2050 2068 2074 2086 2095 2116 2117 2155 2174 2196 2210 2232 2235 2258
2268
## 2321 2323 2330 2336 2382 2391 2393 2403 2409 2424 2429 2472 2477 2479 2486
2492
## 2321 2323 2330 2336 2382 2391 2393 2403 2409 2424 2429 2472 2477 2479 2486
2492
## 2498 2509 2510 2519 2524 2533 2535 2557 2570 2582 2704 2705 2706 2707 2765
2766
## 2498 2509 2510 2519 2524 2533 2535 2557 2570 2582 2704 2705 2706 2707 2765
2766
## 2767 2768 2769 2770 2771 2776 2777 2916 2917 2943 2944 2946 2956 2957 2969
2970
## 2767 2768 2769 2770 2771 2776 2777 2916 2917 2943 2944 2946 2956 2957 2969
2970
## 2971 2988 3014 3034 3035 3047 3048 3057 3058 3062 3063 3068 3071 3073 3075
3077
## 2971 2988 3014 3034 3035 3047 3048 3057 3058 3062 3063 3068 3071 3073 3075
3077
## 3082 3083 3084 3103 3104 3121 3130 3131 3144 3145 3192 3205 3206 3220 3235
3257
## 3082 3083 3084 3103 3104 3121 3130 3131 3144 3145 3192 3205 3206 3220 3235
3257
## 3260 3261 3301 3302 3303 3310 3311 3312 3313 3316 3319 3354 3359 3371 3401
3411
## 3260 3261 3301 3302 3303 3310 3311 3312 3313 3316 3319 3354 3359 3371 3401
3411
## 3431 3493 3494 3495 3496 3498 3512 3517 3518 3528 3538 3547 3550 3553 3560
3563
## 3431 3493 3494 3495 3496 3498 3512 3517 3518 3528 3538 3547 3550 3553 3560
3563
## 3573 3577 3599 3627 3628 3629 3630 3631 3633 3634 3635 3636 3690 3692 3693
3753
## 3573 3577 3599 3627 3628 3629 3630 3631 3633 3634 3635 3636 3690 3692 3693
3753
```

```
## 3754 3778 3796 3805 3808 3818 3824 3859 3873 3919 3920 4006 4026 4029 4030
4031
## 3754 3778 3796 3805 3808 3818 3824 3859 3873 3919 3920 4006 4026 4029 4030
4031
## 4032 4033 4046 4057 4112 4113 4150 4155 4159 4164 4171 4189 4190 4192 4193
4201
## 4032 4033 4046 4057 4112 4113 4150 4155 4159 4164 4171 4189 4190 4192 4193
4201
## 4311 4351 4352 4457 4464 4465 4485 4486 4487 4488 4489 4600 4601 4673 4674
4694
## 4311 4351 4352 4457 4464 4465 4485 4486 4487 4488 4489 4600 4601 4673 4674
4694
## 4741 4745 4746 4747 4748 4756 4771 4772 4794 4795 4796 4797 4816 4817 4818
4819
## 4741 4745 4746 4747 4748 4756 4771 4772 4794 4795 4796 4797 4816 4817 4818
4819
## 4871 4872 4886 4889 4903 4926 4927 4928 4967 4971 4972 5018 5020 5021 5113
5114
## 4871 4872 4886 4889 4903 4926 4927 4928 4967 4971 4972 5018 5020 5021 5113
5114
## 5115 5124 5127 5128 5129 5130 5138 5147 5148 5217 5218 5219 5220 5228 5247
5252
## 5115 5124 5127 5128 5129 5130 5138 5147 5148 5217 5218 5219 5220 5228 5247
5252
## 5267 5268 5286 5287 5316 5331 5334 5335 5337 5339 5340 5347 5358 5376 5404
5405
## 5267 5268 5286 5287 5316 5331 5334 5335 5337 5339 5340 5347 5358 5376 5404
5405
## 5413 5414 5427 5428 5439 5440 5441 5442 5489 5490 5522 5523 5524 5527 5542
5544
## 5413 5414 5427 5428 5439 5440 5441 5442 5489 5490 5522 5523 5524 5527 5542
5544
## 5545 5546 5549 5550 5552 5553 5563 5570 5582 5585 5591 5592 5600 5603 5607
5610
## 5545 5546 5549 5550 5552 5553 5563 5570 5582 5585 5591 5592 5600 5603 5607
5610
## 5611 5616 5624 5625 5628 5634 5637 5638 5640 5643 5645 5647 5652 5653 5668
5671
## 5611 5616 5624 5625 5628 5634 5637 5638 5640 5643 5645 5647 5652 5653 5668
5671
## 5678 5686 5688 5698 5702 5709 5710 5713 5720 5724 5730 5764 5765 5766 5768
5793
## 5678 5686 5688 5698 5702 5709 5710 5713 5720 5724 5730 5764 5765 5766 5768
5793
## 5824 5825 5826 5828 5859 5860 5861 5866 5870 5874 5875 5878 5892 5909 5917
5921
## 5824 5825 5826 5828 5859 5860 5861 5866 5870 5874 5875 5878 5892 5909 5917
5921
## 5930 5933 5937 5938 5948 5949 5950 5951 5954 5990 6007 6012 6013 6015 6045
6046
```

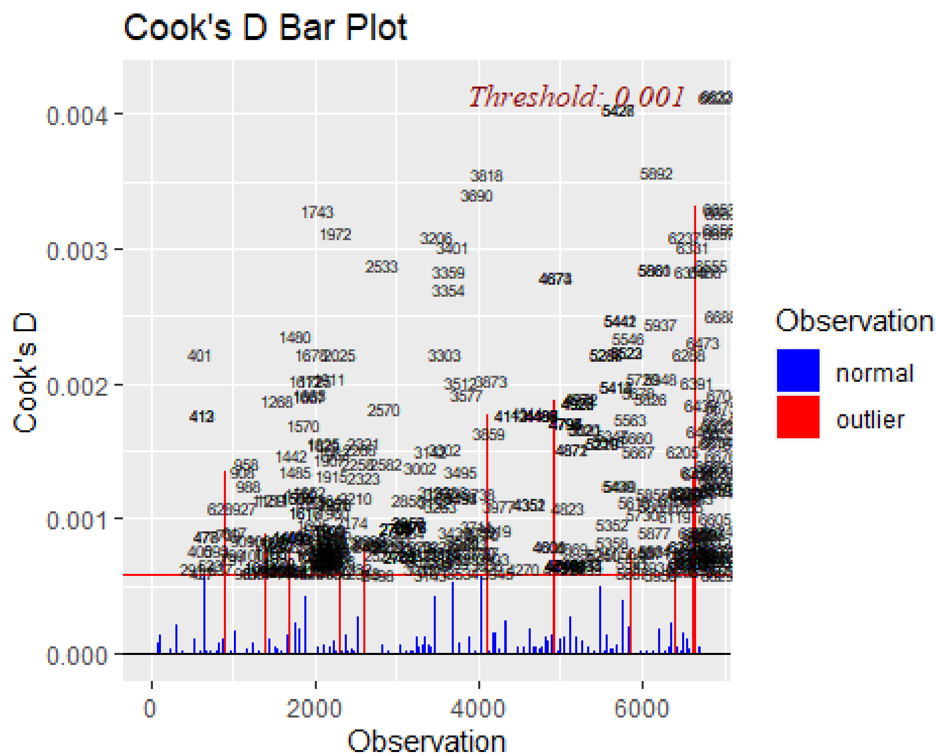
```
## 5930 5933 5937 5938 5948 5949 5950 5951 5954 5990 6007 6012 6013 6015 6045
6046
## 6053 6065 6066 6067 6092 6093 6196 6229 6235 6236 6237 6243 6253 6259 6262
6263
## 6053 6065 6066 6067 6092 6093 6196 6229 6235 6236 6237 6243 6253 6259 6262
6263
## 6287 6288 6292 6294 6298 6299 6318 6323 6331 6340 6348 6383 6384 6385 6387
6391
## 6287 6288 6292 6294 6298 6299 6318 6323 6331 6340 6348 6383 6384 6385 6387
6391
## 6401 6408 6413 6414 6417 6420 6421 6439 6449 6460 6461 6465 6466 6473 6482
6486
## 6401 6408 6413 6414 6417 6420 6421 6439 6449 6460 6461 6465 6466 6473 6482
6486
## 6490 6500 6501 6508 6509 6510 6511 6512 6521 6522 6535 6536 6542 6543 6552
6555
## 6490 6500 6501 6508 6509 6510 6511 6512 6521 6522 6535 6536 6542 6543 6552
6555
## 6558 6559 6560 6561 6562 6569 6603 6616 6622 6626 6628 6633 6634 6635 6639
6644
## 6558 6559 6560 6561 6562 6569 6603 6616 6622 6626 6628 6633 6634 6635 6639
6644
## 6653 6654 6655 6656 6657 6676 6683 6686 6688 6690 6699 6701 6704 6720 6730
6731
## 6653 6654 6655 6656 6657 6676 6683 6686 6688 6690 6699 6701 6704 6720 6730
6731

###There are a large amount of cases that have been identified as High levera
ge
plot(dffits(md2)) # Compare to 2sqrt(p/n) for large datasets and 1 for small
```



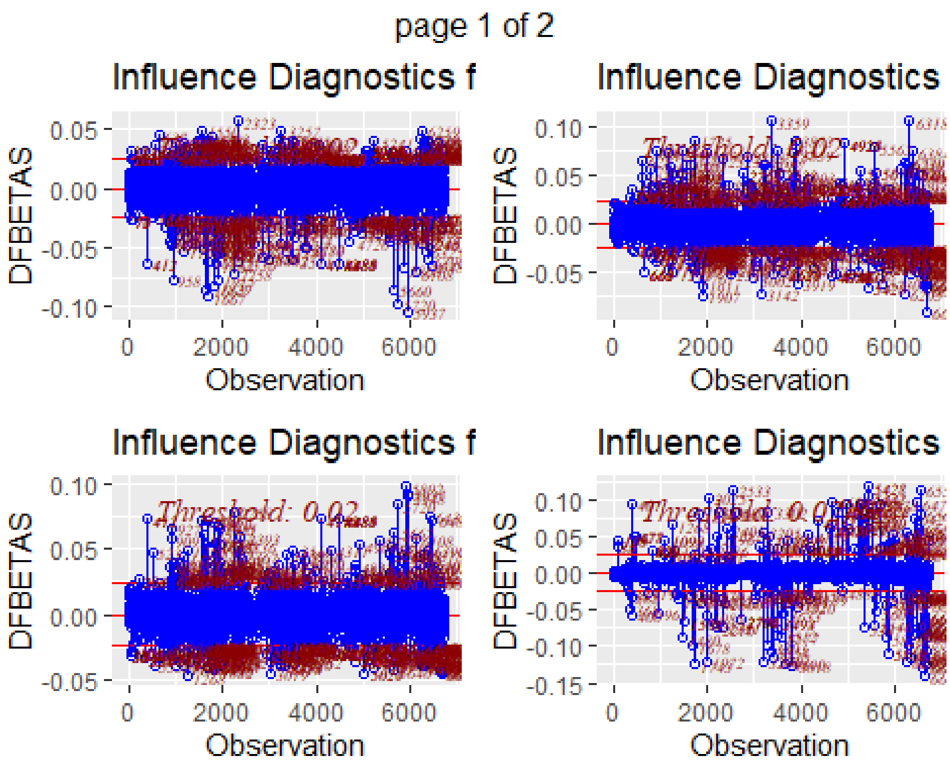

```
which(dffits(md2)>1)

## named integer(0)

which(dfbetas(md2)>1) # Compare to 2/sqrt(n) for large datasets and 1 for sma
ll

## integer(0)

plot(cooks.distance(md2)) # Compare percentile F(p,n-p) to 10th or 20th
```

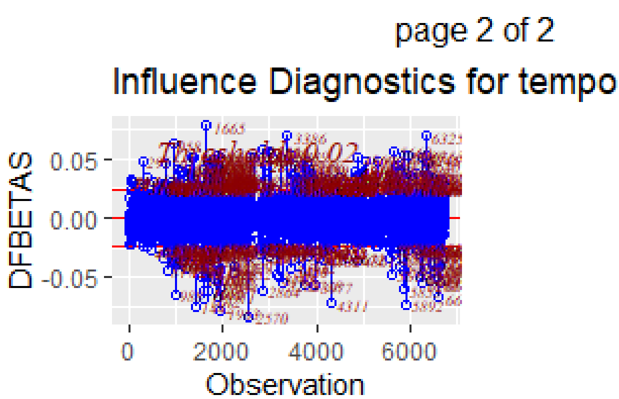



```
q <- pf(cooks.distance(md2),5,6746-5)
which(q>.1)

## named integer(0)

which(q>.2)

## named integer(0)

ols_plot_cooksd_bar(md2) # One way to visualize Cook's distance
```

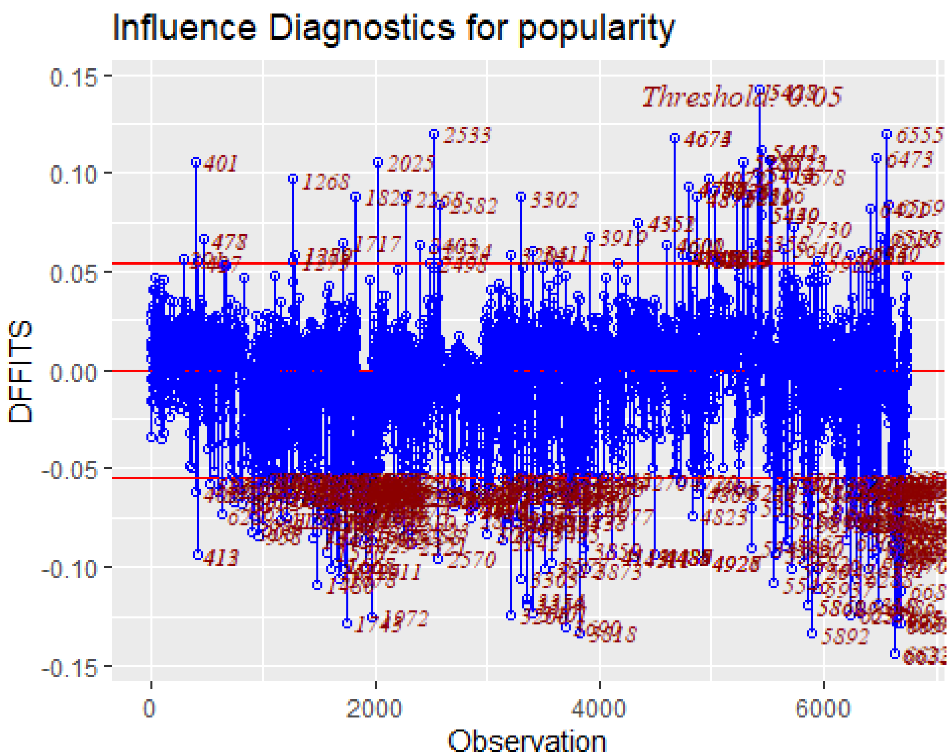



```
ols_plot_dfbetas(md2) # Visualize influence on estimation of betas
```






```r
ols_plot_dffits(md2) # Visualize influence on estimation of Y
```



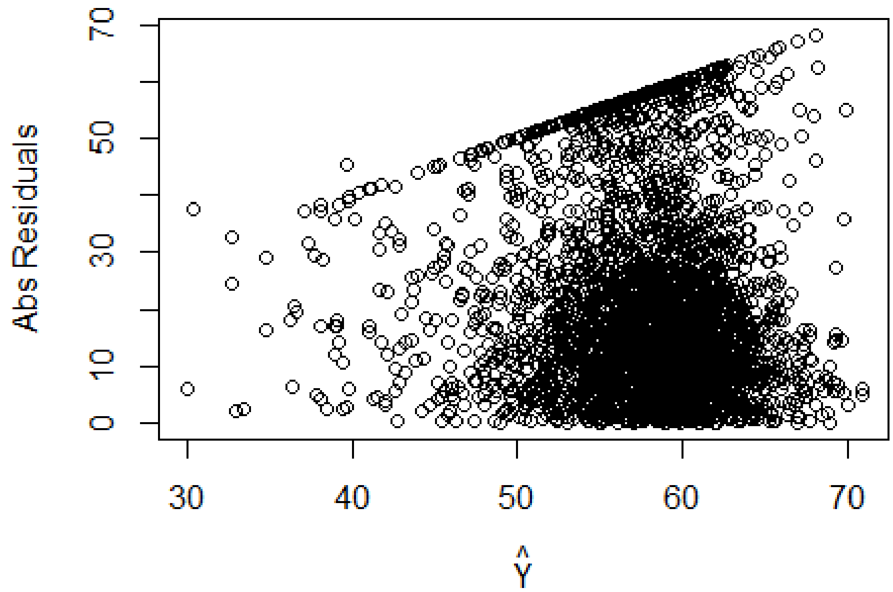

```r
mdl.out = lm(popularity ~ danceability + loudness + instrumentalness, subset
= -c(6657,6676, 6683, 6686, 6688, 6690, 6699, 6701, 6704, 6720, 6730, 6731, 6
622, 6626, 6628, 6633, 6634, 6635, 6639, 6644, 6653, 6654, 6655, 6656,36, 84,
85, 90, 99, 100, 103, 104, 105, 106, 108,109, 183, 188, 189, 190, 201, 222, 2
23, 228, 304, 305, 306, 320, 321, 328, 329, 345, 368,385, 400, 401, 412, 413,
427, 429, 477, 478, 497, 499, 511, 648, 659, 660, 661, 662, 663, 667, 702, 74
0, 766, 767, 768, 853, 855, 908, 961, 1080, 1084, 1098, 1099, 1159, 1204, 120
5, 1206, 1207, 1211, 1264, 1268, 1273, 1279, 1280, 1315, 1327, 1331, 1349, 13
53, 1450, 1478, 1480, 1485, 1504, 1505, 1523, 1550, 1570, 1572, 1574, 1575, 1
576, 1678,1584, 1596, 1612, 1616, 1617, 1641, 1651, 1654, 1665, 1678,1681, 17
04, 1717, 1721, 1725, 1743, 1745, 1771, 1772, 1797, 1799, 1800, 1801, 1804, 1
805, 1820, 1825, 1911, 1951, 1963, 1972, 1976, 1991, 2003, 2004, 2025, 2038,
2050, 2068, 2074, 2086, 2095, 2116, 2117, 2155, 2174, 2196, 2210, 2232, 2235,
2258, 2268, 2321, 2323, 2330, 2336, 2382, 2391, 2393, 2403, 2409, 2424, 2429,
2472, 2477, 2479, 2486, 2492, 2498, 2509, 2510, 2519, 2524, 2533, 2535, 2557,
2570, 2582, 2704, 2705, 2706, 2707, 2765, 2766, 2767, 2768, 27669, 2770, 2771
, 2776, 2777, 2916, 2917, 2943, 2944, 2946, 2956, 2957, 2969, 2970, 2971, 298
8, 3014, 3034))
```

3. Constant Variance

There are no apparent issues with non-constant variance.

```
```r
```
```

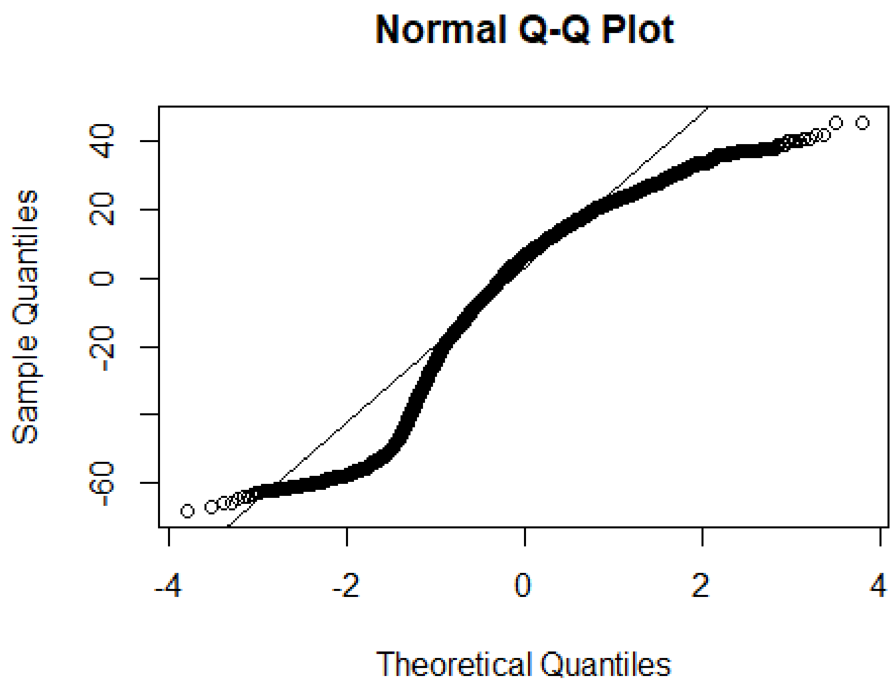
```
plot(abs(residuals(md2))~predict(md2), xlab = expression(hat(Y)), ylab = "Abs Residuals")
```



4. Normality

A Q-Q plot supports approximate normality.

```
qqnorm(residuals(md2))
qqline(residuals(md2))
```


Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
40
20
0
-20
-60
-4
-2
0
2
4