# main

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Narative:

What musical characteristics of an electronic song make it popular? This is the question I aim to answer in my project.

By characteristics of a song, I mean the basic features in the context of music theory such as tempo, key, time signature, etc. But I also mean more complex features in the context of how humans interpret music such as, loudness, energy, danceability, etc.

By popularity of a song, I mean how many times that song was listened to compared to other songs.

For music producers, an accurate answer to this question would be very valuable as it would give them evidence for selecting target characteristics to maximize a songs popularity. It would also give them target characteristics for a certain type of song. For example, if a music producer wanted to create a high energy and very danceable song, they would have evidence as to which key, time signature, tempo, etc. they would need to maximize the songs popularity.

My response variable would be the popularity of a song based on its musical characteristics. However, my dataset only contains time-series measures of popularity. That is, my dataset only contains information of how popular a song is currently but I am more interested in how popular a song is independent of when it was uploaded. The songs popularity in the dataset is also dependent on the popularity of the artists from the assumption that more popular artists have bigger platforms to promote their song. I would like my response variable to also be independent of the popularity of the artists. Therefor I need to construct a custom response variable.

To tackle the issue of the current popularity metric from the data-set being a time dependent variable. The response variable would be the time dependent popularity metric but it's predicted value would be calculated based on a given time and artist popularity.

This article :https://towardsdatascience.com/song-popularity-predictor-1ef69735e380 did a very similar thing to my project. Only it used the song popularity as a predictor variable to estimate the song features. That is, it analyzed the distribution of popular songs to predict the features of those songs.

My hypothesis is that a songs popularity is dependedent on its musical characteristics.

Data:

This dataset is retreived from Spotify's API. See the script below.

Script: https://colab.research.google.com/drive/1VIpipujOuut-qT1iVkMw8lASIqxjEGDv?usp=sharing

The dataset was created by selecting playlists within the Electronic Music Genre, then calling the API to retreive track information for every track within every playlist. Then for every track I called the API again to get information that was not included in the first call. This is information like artist information and upload date. I then parsed everything into a dictionary and wrote it to the CSV file.

I may need to do some more data augmentation. For example, I may want to augment some of the predictor variables I imagine to have no linear relationship with the response based on the context of the problem to be categorical to account for this non-linear relationship.

This article: https://towardsdatascience.com/song-popularity-predictor-1ef69735e380 used logistic regression which is evidence that categorical variables may be more accurate predictors. It also did a fair bit more cleaning of the dataset. Which is evidence that I may need to look at my data more closely for corrupted entries.

```
track_data <- read.csv('data/spotify_tracks.csv')
```

```
glimpse(track_data)
```

```
## Rows: 2,121
## Columns: 25
## $ track.name       <chr> "Move Your Body", "Say Nothing (feat. MAY-A)", "You'v~
## $ artists.names    <chr> "['Ã-wnboss', 'Sevek']", "['Flume', 'MAY-A']", "['Dis~
## $ artists.ids      <chr> "['37czgDRfGMvgRiUKHvnnhj', '0aOIluXr131XqrXFwFCFGT']~
## $ track.popularity <int> 79, 69, 72, 85, 79, 51, 83, 52, 76, 75, 86, 63, 69, 6~
## $ artist.ids       <chr> "['37czgDRfGMvgRiUKHvnnhj', '0aOIluXr131XqrXFwFCFGT']~
## $ track.id         <chr> "6GomT970rCOkKAyyrwJeZi", "424Uwmm1kNW07Ty1nOhSpl", "~
## $ release.date     <chr> "2021-10-29", "2022-02-02", "2022-01-28", "2021-11-19~
## $ danceability     <dbl> 0.848, 0.478, 0.658, 0.308, 0.788, 0.774, 0.601, 0.73~
## $ energy           <dbl> 0.821, 0.822, 0.908, 0.861, 0.945, 0.784, 0.787, 0.90~
## $ key              <int> 2, 2, 4, 11, 9, 10, 0, 10, 9, 11, 0, 10, 3, 11, 10, 7~
## $ loudness         <dbl> -5.408, -1.961, -8.071, -4.112, -5.091, -4.549, -6.17~
## $ mode             <int> 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,~
## $ speechiness      <dbl> 0.0527, 0.0642, 0.1860, 0.1720, 0.0599, 0.0351, 0.031~
## $ acousticness     <dbl> 0.016900, 0.101000, 0.033500, 0.012000, 0.223000, 0.1~
## $ instrumentalness <dbl> 4.03e-04, 6.60e-05, 1.98e-02, 1.03e-03, 2.97e-06, 1.0~
## $ liveness         <dbl> 0.0962, 0.1200, 0.2210, 0.2770, 0.1150, 0.2930, 0.142~
## $ valence          <dbl> 0.2490, 0.3090, 0.4500, 0.0386, 0.4660, 0.5330, 0.525~
## $ tempo            <dbl> 125.051, 130.058, 123.920, 171.966, 128.036, 127.042,~
## $ type             <chr> "audio_features", "audio_features", "audio_features",~
## $ id               <chr> "6GomT970rCOkKAyyrwJeZi", "424Uwmm1kNW07Ty1nOhSpl", "~
## $ duration_ms      <int> 157445, 232959, 253787, 158774, 168053, 177689, 17516~
## $ time_signature   <int> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4,~
## $ follow           <int> 33201, 2257049, 7669295, 125502, 3198289, 1251567, 18~
## $ pop              <int> 130, 135, 158, 146, 150, 153, 85, 140, 74, 297, 168, ~
## $ num.artist       <int> 2, 2, 2, 2, 2, 2, 1, 2, 1, 4, 2, 2, 4, 1, 1, 2, 1, 3,~
```

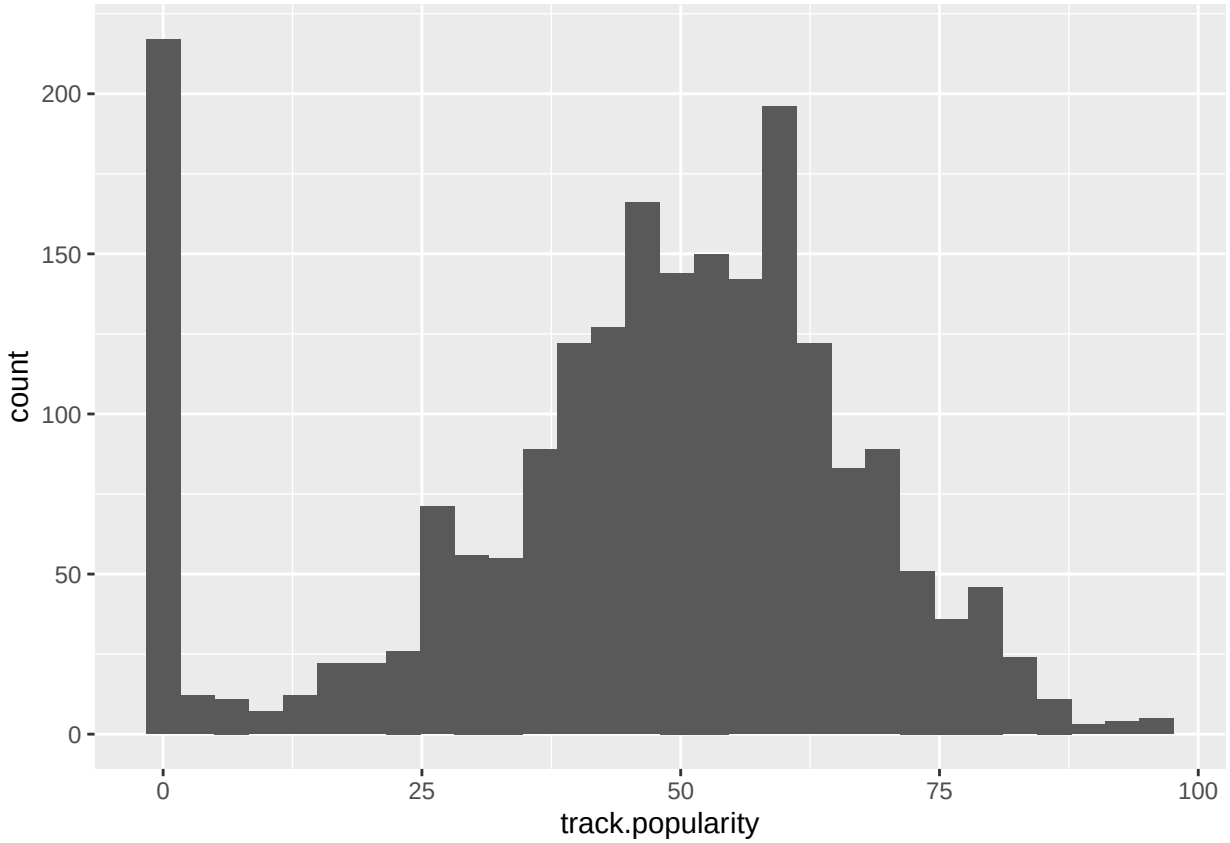
```
summary(track_data)
```

```
##   track.name        artists.names      artists.ids        track.popularity
##  Length:2121        Length:2121        Length:2121        Min.   : 0.0
```

```
##  Class :character   Class :character   Class :character   1st Qu.:36.0
##  Mode  :character   Mode  :character   Mode  :character   Median :49.0
##                                                          Mean   :45.6
##                                                          3rd Qu.:60.0
##                                                          Max.   :96.0
##   artist.ids          track.id         release.date        danceability
##  Length:2121        Length:2121        Length:2121        Min.   :0.1310
##  Class :character   Class :character   Class :character   1st Qu.:0.5730
##  Mode  :character   Mode  :character   Mode  :character   Median :0.6690
##                                                          Mean   :0.6546
##                                                          3rd Qu.:0.7490
##                                                          Max.   :0.9680
##      energy            key            loudness            mode
##  Min.   :0.0545   Min.   : 0.000   Min.   :-23.048   Min.   :0.0000
##  1st Qu.:0.6690   1st Qu.: 2.000   1st Qu.: -8.176   1st Qu.:0.0000
##  Median :0.7930   Median : 6.000   Median : -6.244   Median :0.0000
##  Mean   :0.7567   Mean   : 5.553   Mean   : -6.689   Mean   :0.4828
##  3rd Qu.:0.8810   3rd Qu.: 9.000   3rd Qu.: -4.712   3rd Qu.:1.0000
##  Max.   :0.9990   Max.   :11.000   Max.   :  0.385   Max.   :1.0000
##   speechiness       acousticness       instrumentalness       liveness
##  Min.   :0.02400   Min.   :0.0000025   Min.   :0.0000000   Min.   :0.0222
##  1st Qu.:0.03900   1st Qu.:0.0058900   1st Qu.:0.0000554   1st Qu.:0.0901
##  Median :0.05010   Median :0.0301000   Median :0.0234000   Median :0.1210
##  Mean   :0.07517   Mean   :0.1218216   Mean   :0.2723508   Mean   :0.1848
##  3rd Qu.:0.07730   3rd Qu.:0.1360000   3rd Qu.:0.6150000   3rd Qu.:0.2370
##  Max.   :0.86900   Max.   :0.9910000   Max.   :0.9820000   Max.   :0.9740
##     valence           tempo            type                id
##  Min.   :0.0174   Min.   : 65.57   Length:2121        Length:2121
##  1st Qu.:0.2060   1st Qu.:122.00   Class :character   Class :character
##  Median :0.3820   Median :125.22   Mode  :character   Mode  :character
##  Mean   :0.4015   Mean   :125.69
##  3rd Qu.:0.5630   3rd Qu.:128.05
##  Max.   :0.9770   Max.   :216.08
##   duration_ms      time_signature      follow               pop
##  Min.   :  92267   Min.   :1.000   Min.   :        3   Min.   :  0.0
##  1st Qu.: 180800   1st Qu.:4.000   1st Qu.:    27959   1st Qu.: 59.0
##  Median : 209077   Median :4.000   Median :   183996   Median : 97.0
##  Mean   : 228598   Mean   :3.982   Mean   :  3447113   Mean   :105.3
##  3rd Qu.: 246092   3rd Qu.:4.000   3rd Qu.:  1713802   3rd Qu.:143.0
##  Max.   :1008533   Max.   :5.000   Max.   :100958599   Max.   :773.0
##    num.artist
##  Min.   : 1.0
##  1st Qu.: 1.0
##  Median : 2.0
##  Mean   : 1.8
##  3rd Qu.: 2.0
##  Max.   :14.0
```

```
t_popularity <- ggplot(track_data, aes(x=track.popularity)) + geom_histogram()
t_popularity
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

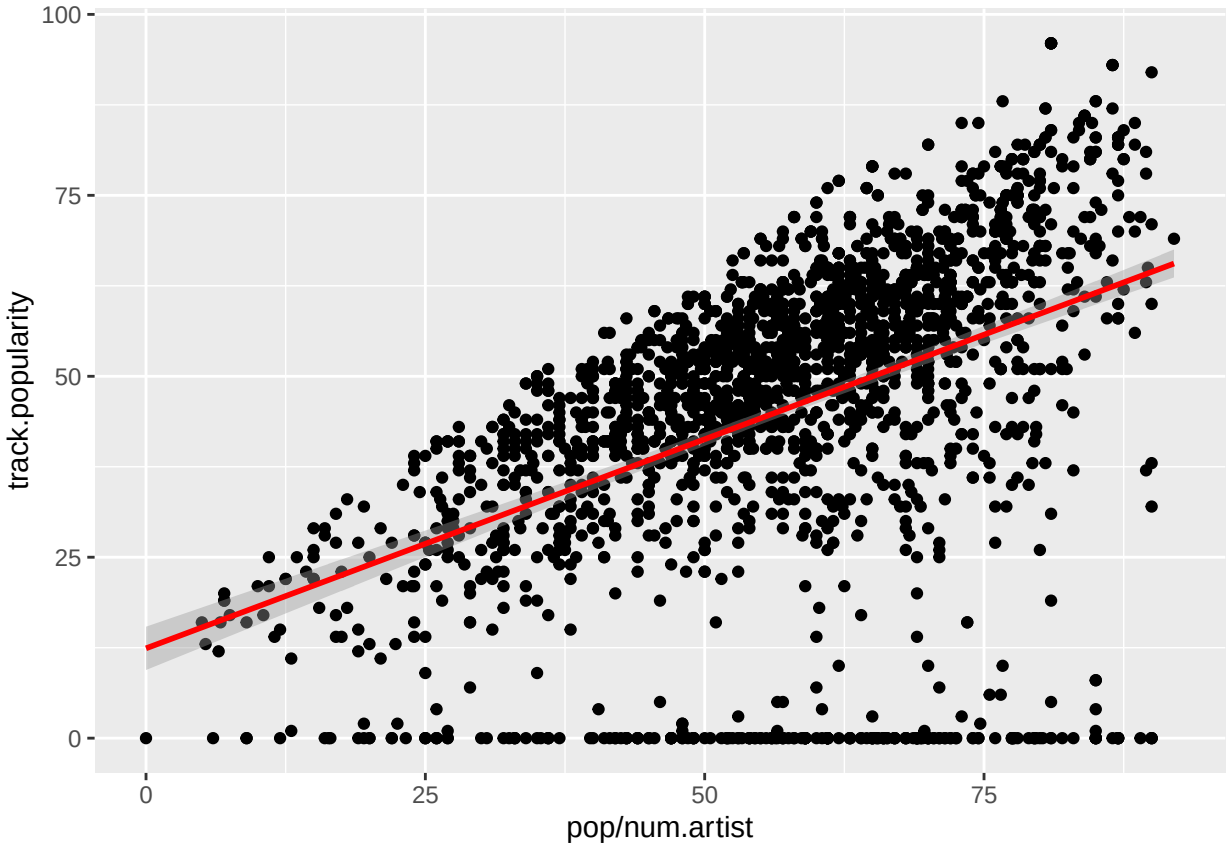



The track popularity seems very normal, asside from the outliers at 0 popularity. Maybe I can change my problem with the given that the track popularity will not be 0? This would allow me to remove such outliers.

Lets see if there may be an ascociation between track popularity and the average artist popularity.

```
popart <- ggplot(data = track_data, aes(x = pop/num.artist , y = track.popularity)) + geom_point()+ sta
popart
```

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




As expected, we can observe a possible association between the popularity of tracks and their artists. We may need to investigate the outlier tracks with popularity is 0.

Lets observe some of the interesting cases, when the popularity of the artists are high but the track popularity is 0.

```
ztpop <- track_data %>% filter(track.popularity == 0) %>% filter((pop/num.artist) > 75)
glimpse(ztpop)
```

```
## Rows: 34
## Columns: 25
## $ track.name       <chr> "Love Is Gone - Fred Rister & Joachim Garraud Radio E~
## $ artists.names    <chr> "['David Guetta']", "['TiÃ«sto']", "['Sia']", "['TiÃ«~
## $ artists.ids      <chr> "['1Cs0zKBU1kc0i8ypK3B9ai']", "['2o5jDhtHVPhrJdv3cEQ9~
## $ track.popularity <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ artist.ids       <chr> "['1Cs0zKBU1kc0i8ypK3B9ai']", "['2o5jDhtHVPhrJdv3cEQ9~
## $ track.id         <chr> "4V9HEnprK5MfCGL8bHHy7y", "6pqFWRuybCtxerWC7B4RgF", "~
## $ release.date     <chr> "2007", "2009-07-22", "2008", "2001", "2015-11-27", "~
## $ danceability     <dbl> 0.788, 0.714, 0.756, 0.609, 0.745, 0.604, 0.707, 0.61~
## $ energy           <dbl> 0.682, 0.642, 0.505, 0.928, 0.680, 0.939, 0.475, 0.48~
## $ key              <int> 4, 8, 11, 8, 11, 10, 9, 10, 9, 0, 0, 1, 6, 0, 7, 7, 0~
## $ loudness         <dbl> -7.826, -10.241, -12.567, -8.597, -6.207, -7.490, -7.~
## $ mode             <int> 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,~
## $ speechiness      <dbl> 0.0694, 0.0868, 0.0804, 0.1220, 0.0508, 0.0544, 0.035~
## $ acousticness     <dbl> 0.01460, 0.00251, 0.00412, 0.06000, 0.01300, 0.01020,~
## $ instrumentalness <dbl> 2.20e-04, 8.87e-01, 8.81e-01, 8.15e-01, 6.13e-02, 6.9~
```

```
## $ liveness         <dbl> 0.0757, 0.1860, 0.4710, 0.0950, 0.0637, 0.0723, 0.172~
## $ valence          <dbl> 0.675, 0.102, 0.103, 0.225, 0.961, 0.204, 0.335, 0.20~
## $ tempo            <dbl> 127.960, 137.834, 128.066, 137.034, 129.991, 139.975,~
## $ type             <chr> "audio_features", "audio_features", "audio_features",~
## $ id               <chr> "4V9HEnprK5MfCGL8bHHy7y", "6pqFWRuybCtxerWC7B4RgF", "~
## $ duration_ms      <int> 201413, 177027, 180133, 173920, 281516, 443427, 22064~
## $ time_signature   <int> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,~
## $ follow           <int> 23811031, 6182612, 22268189, 6182612, 5592939, 618261~
## $ pop              <int> 90, 87, 89, 87, 155, 87, 90, 87, 159, 155, 180, 85, 7~
## $ num.artist       <int> 1, 1, 1, 1, 2, 1, 1, 1, 2, 2, 2, 1, 1, 2, 1, 3, 1, 1,~
```

Lets observe the song with track id = 4V9HEnprK5MfCGL8bHHy7y in this new data frame. The song url is https://open.spotify.com/track/4V9HEnprK5MfCGL8bHHy7y

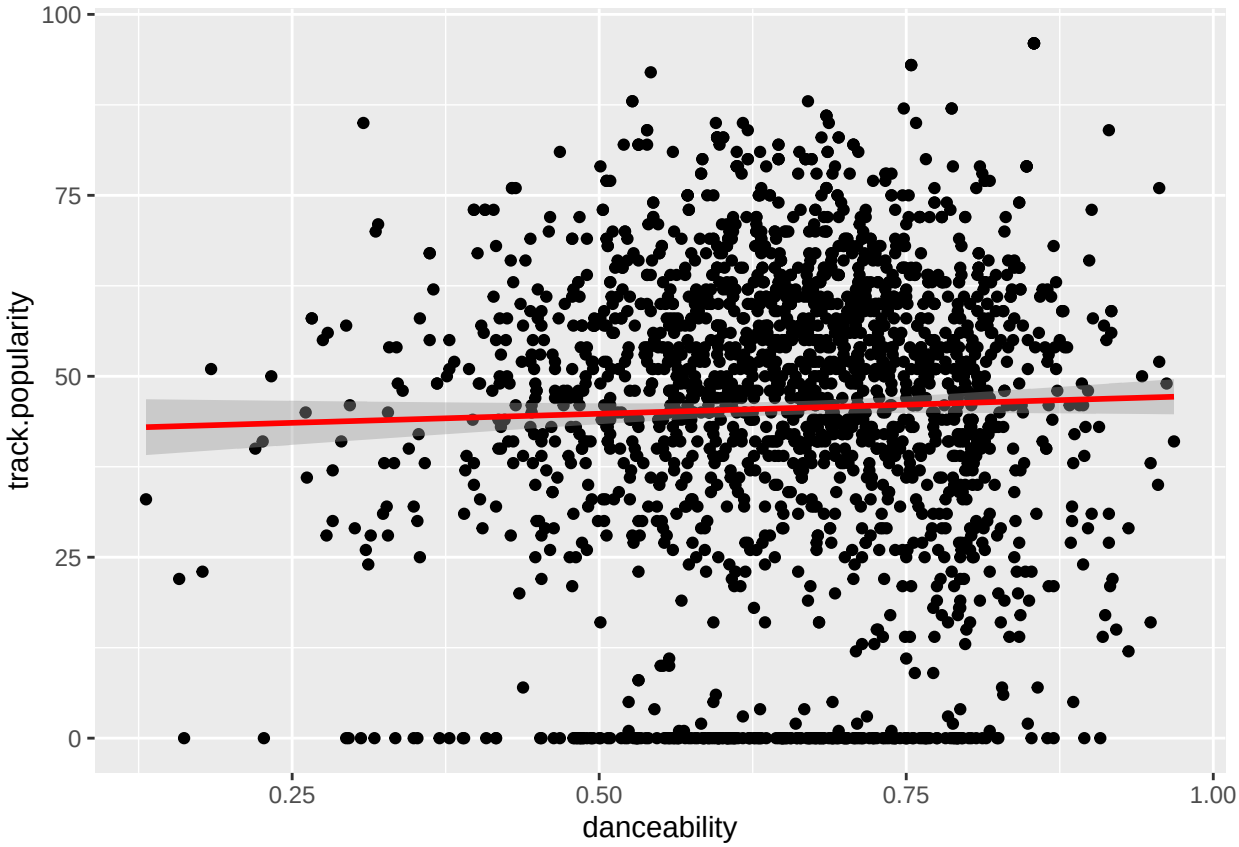
We can see that this song is identical to a song with a different track id of 2MGRnjJc7C0z3EOEWRqcMw and url of https://open.spotify.com/track/2MGRnjJc7C0z3EOEWRqcMw.

It appears that the first track id is an older publication of the same track. And the second track id is a new publication. The new publication has a track popularity of 63. This suggest that the track entry in my dataset is a corrupted entry because it does not accuratley represent the popularity of a song with given features.

Lets examine the continuous audio features of a track and the track popularity.

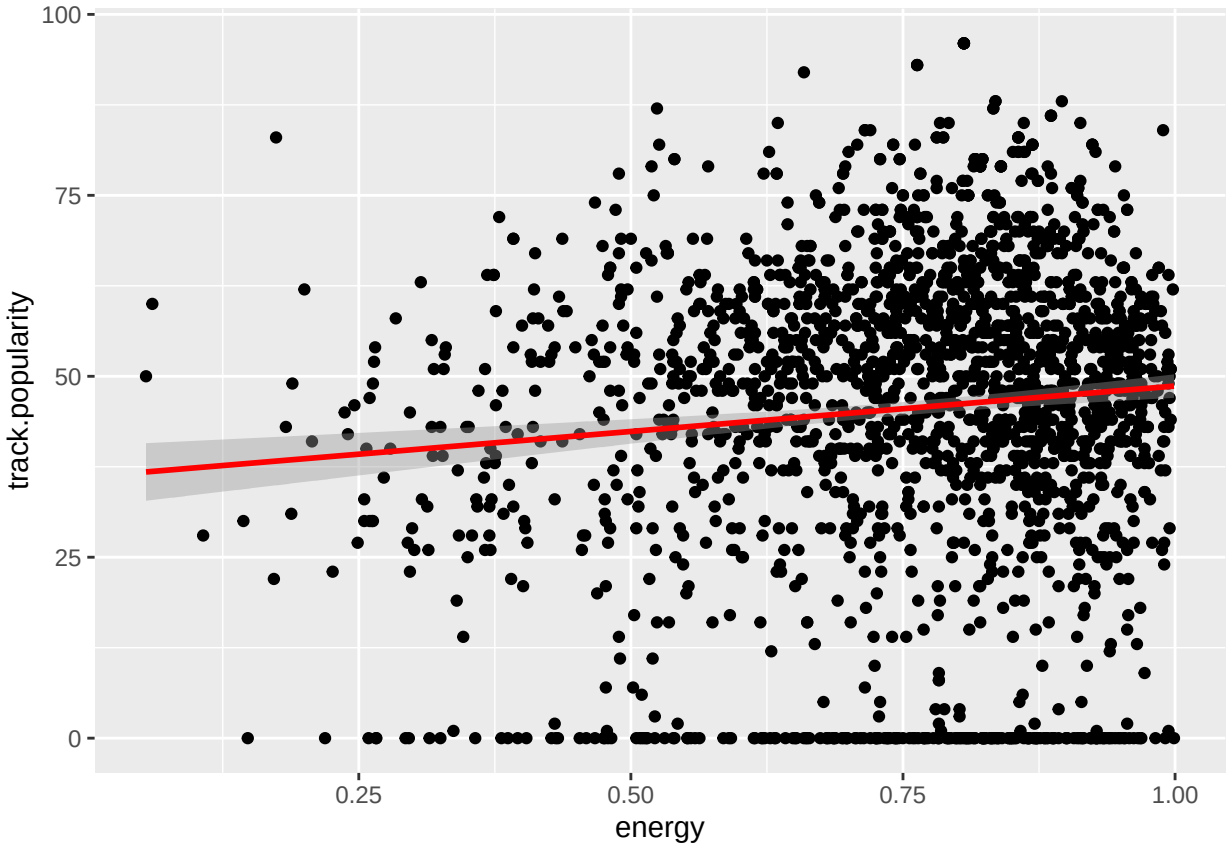
```
pdance <- ggplot(data = track_data, aes(x = danceability , y = track.popularity)) + geom_point()+ stat_s

penergy <- ggplot(data = track_data, aes(x = energy , y = track.popularity)) + geom_point()+ stat_smoothl

ploud <- ggplot(data = track_data, aes(x = loudness , y = track.popularity)) + geom_point()+ stat_smoothl

pspeech <- ggplot(data = track_data, aes(x = speechiness , y = track.popularity)) + geom_point()+ stat_s

pacoustic <- ggplot(data = track_data, aes(x = acousticness , y = track.popularity)) + geom_point()+ sta

pinstrum <- ggplot(data = track_data, aes(x = instrumentalness , y = track.popularity)) + geom_point()+

popart <- ggplot(data = track_data, aes(x = liveness , y = track.popularity)) + geom_point()+ stat_smoot

pvalence <- ggplot(data = track_data, aes(x = valence , y = track.popularity)) + geom_point()+ stat_smoo

ptempo <- ggplot(data = track_data, aes(x = tempo , y = track.popularity)) + geom_point()+ stat_smooth(m

pdance
```

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



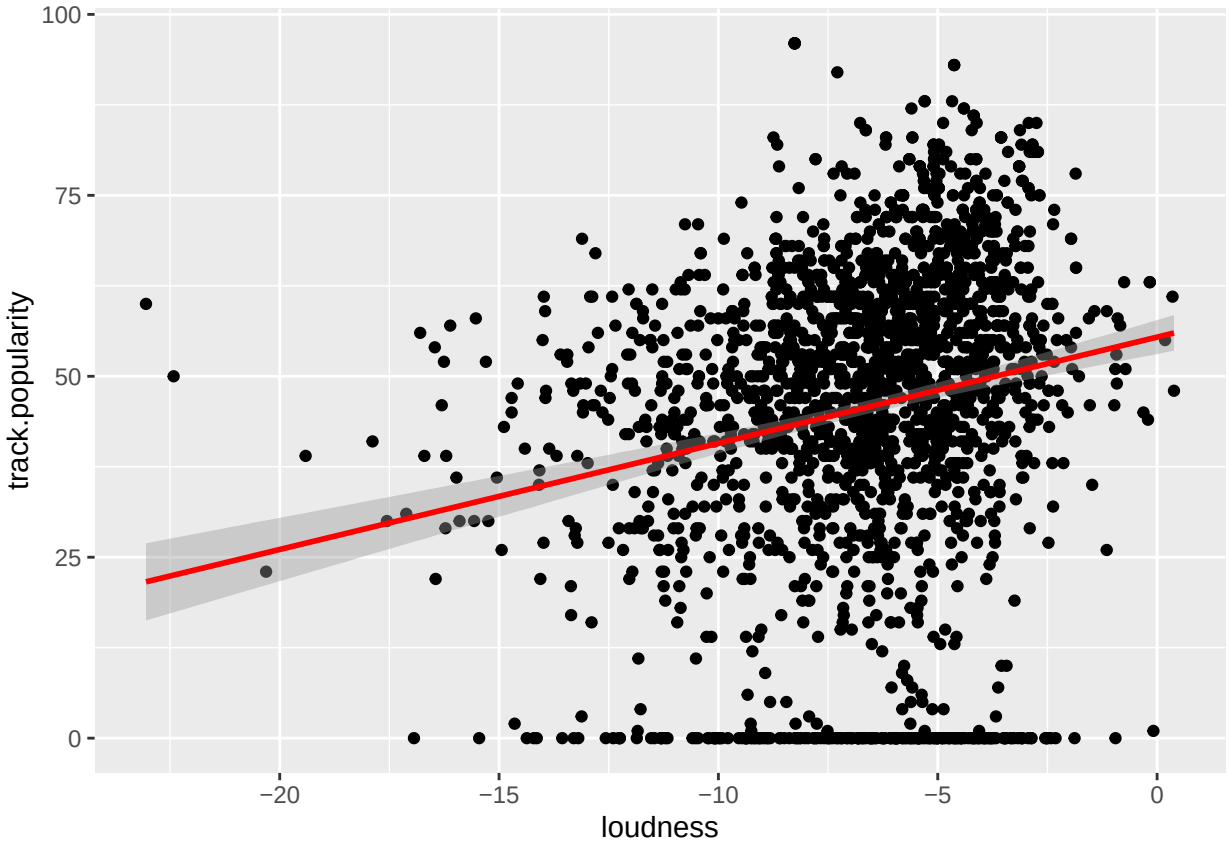

```
penergy
```

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



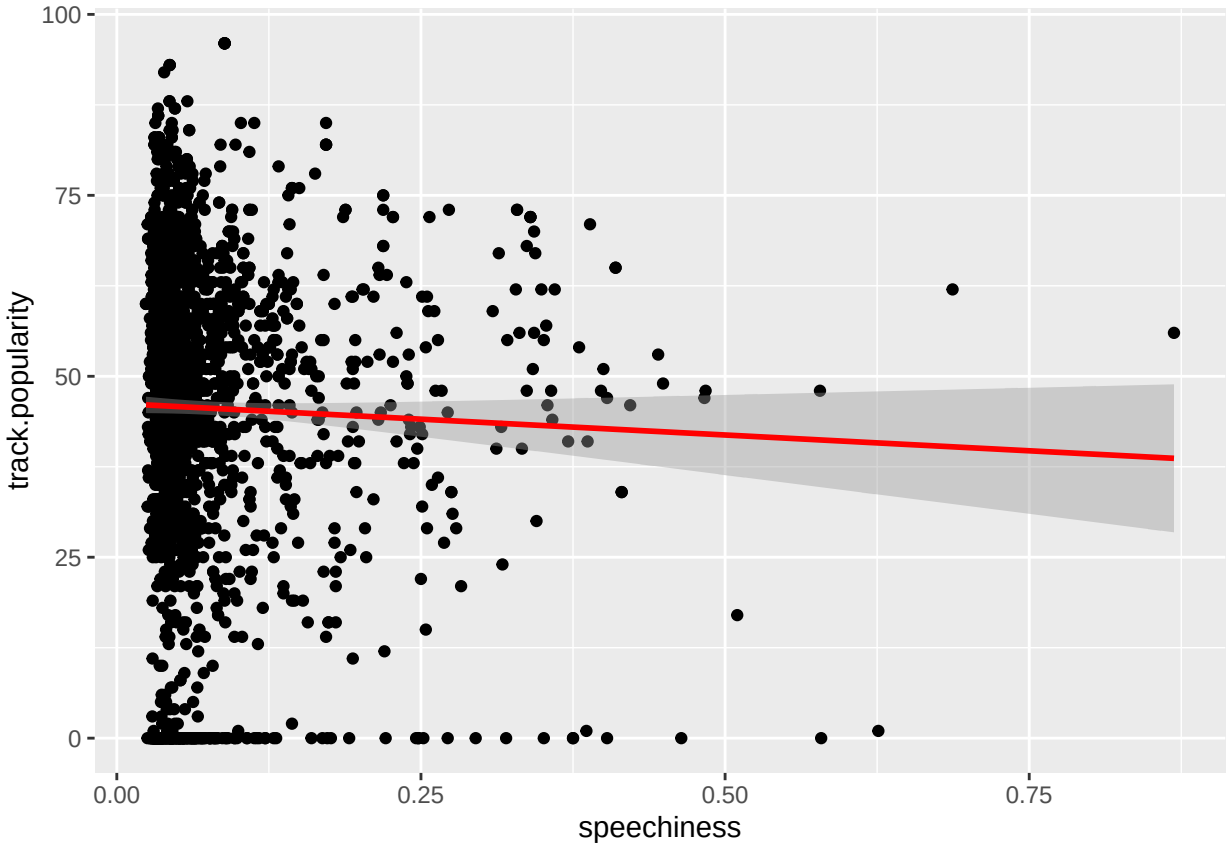

```
ploud
```

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



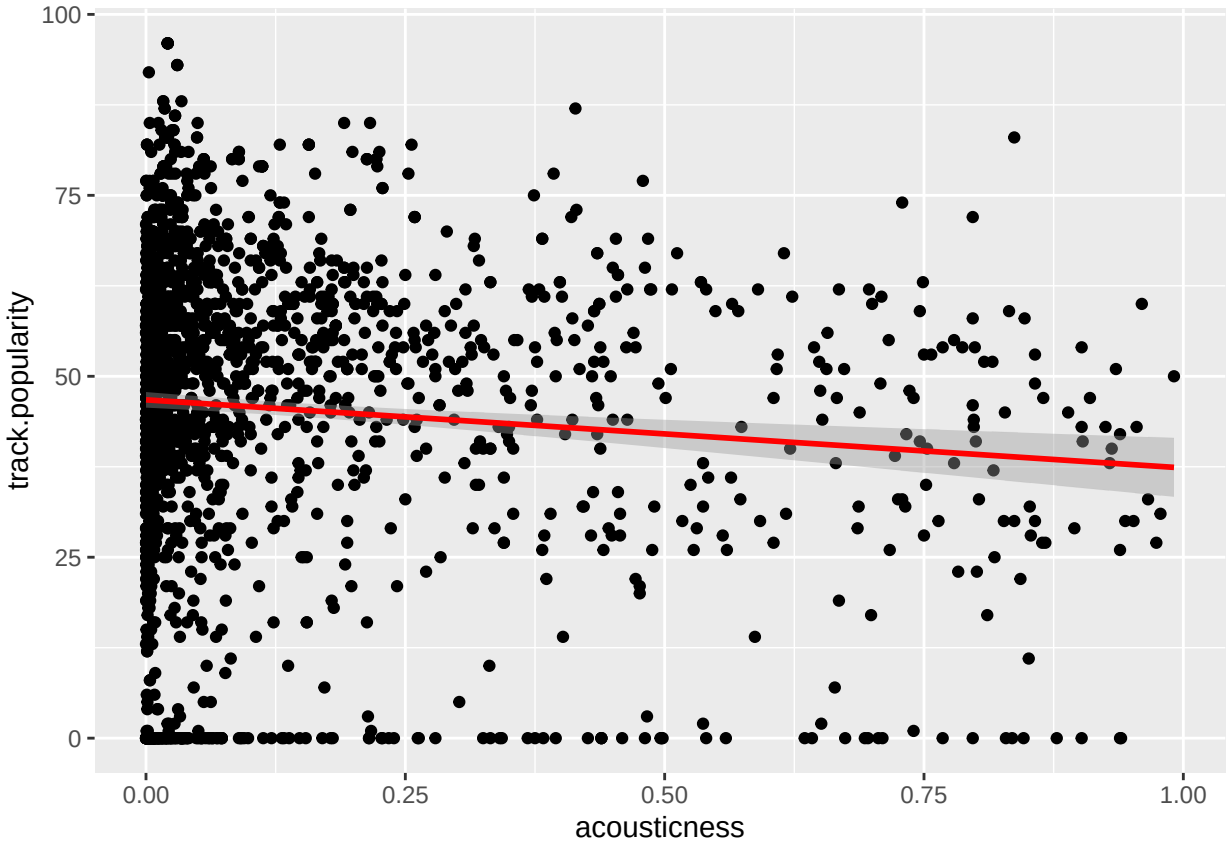

```
pspeech
```

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



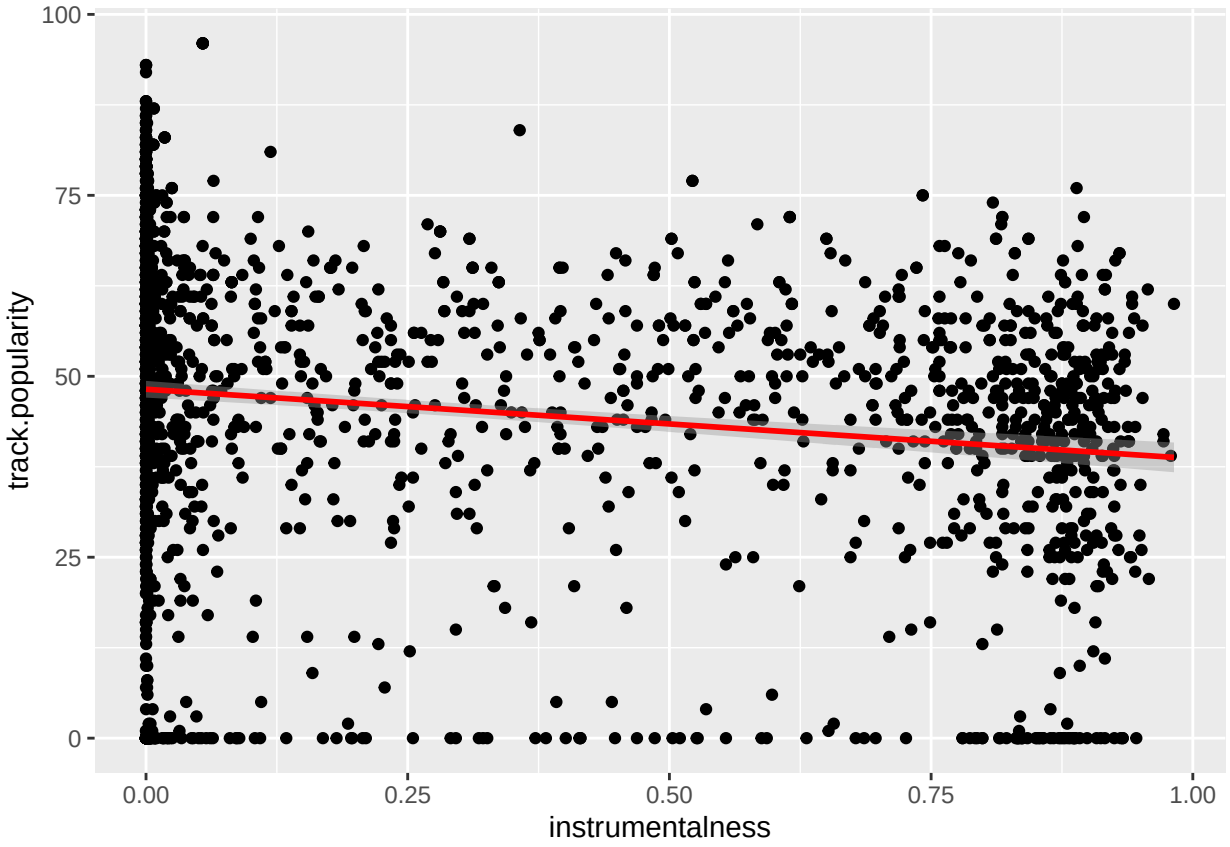

```
pacoustic
```

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



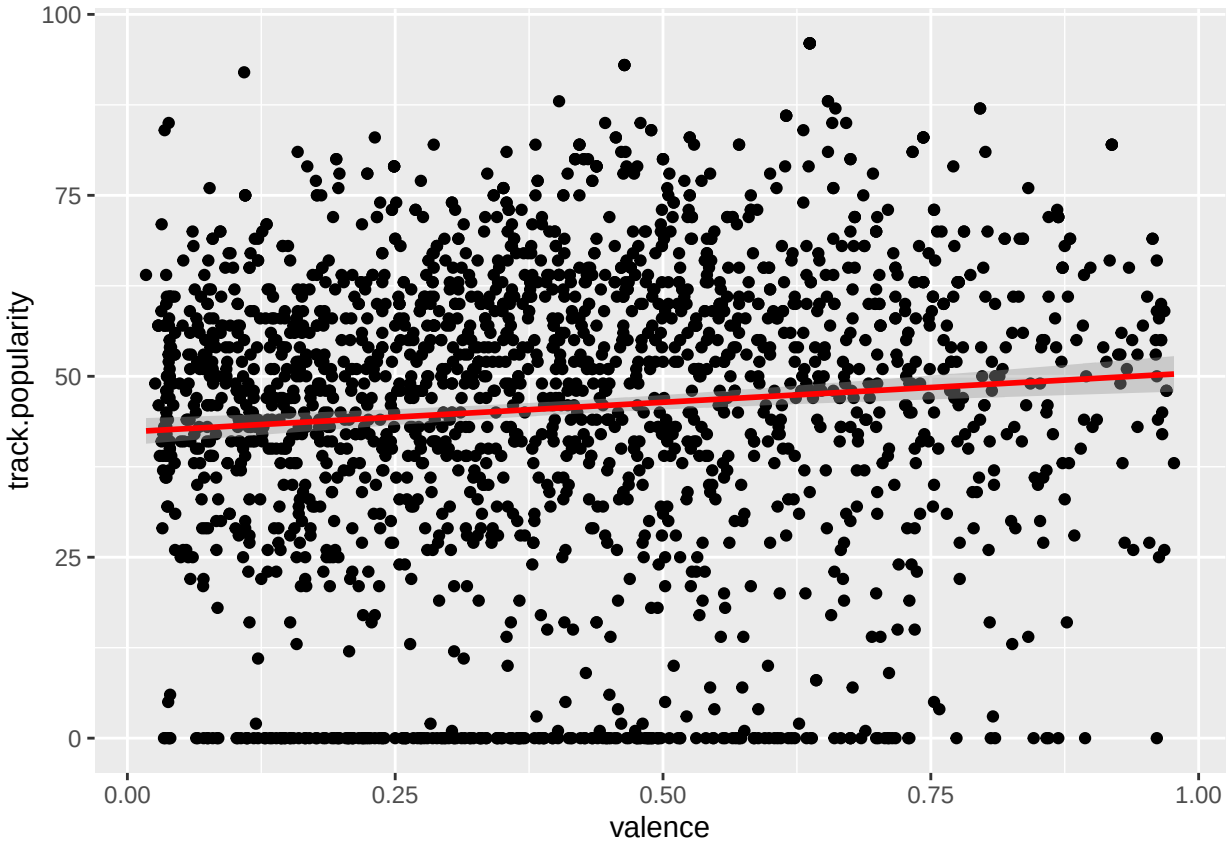

```
pinstrum
```

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



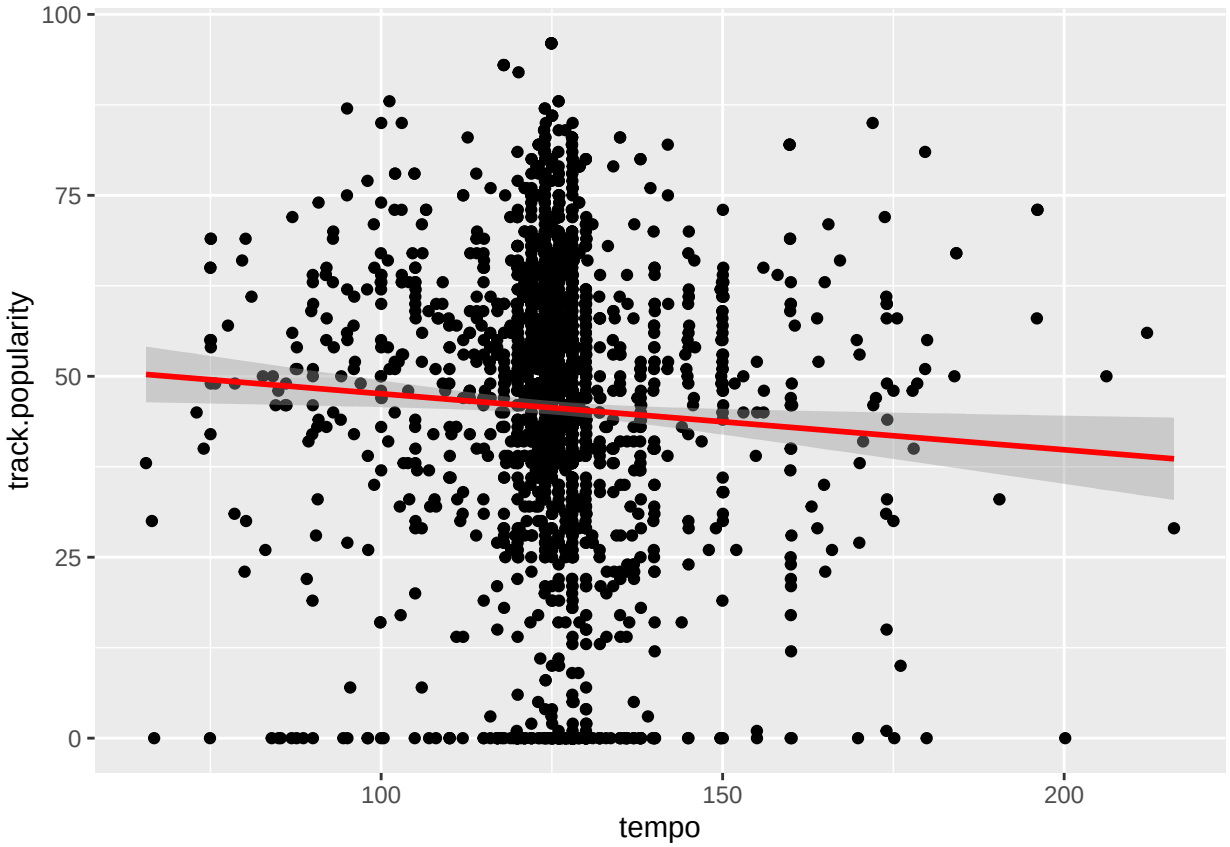

```
pvalence
```

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




```
ptempo
```

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



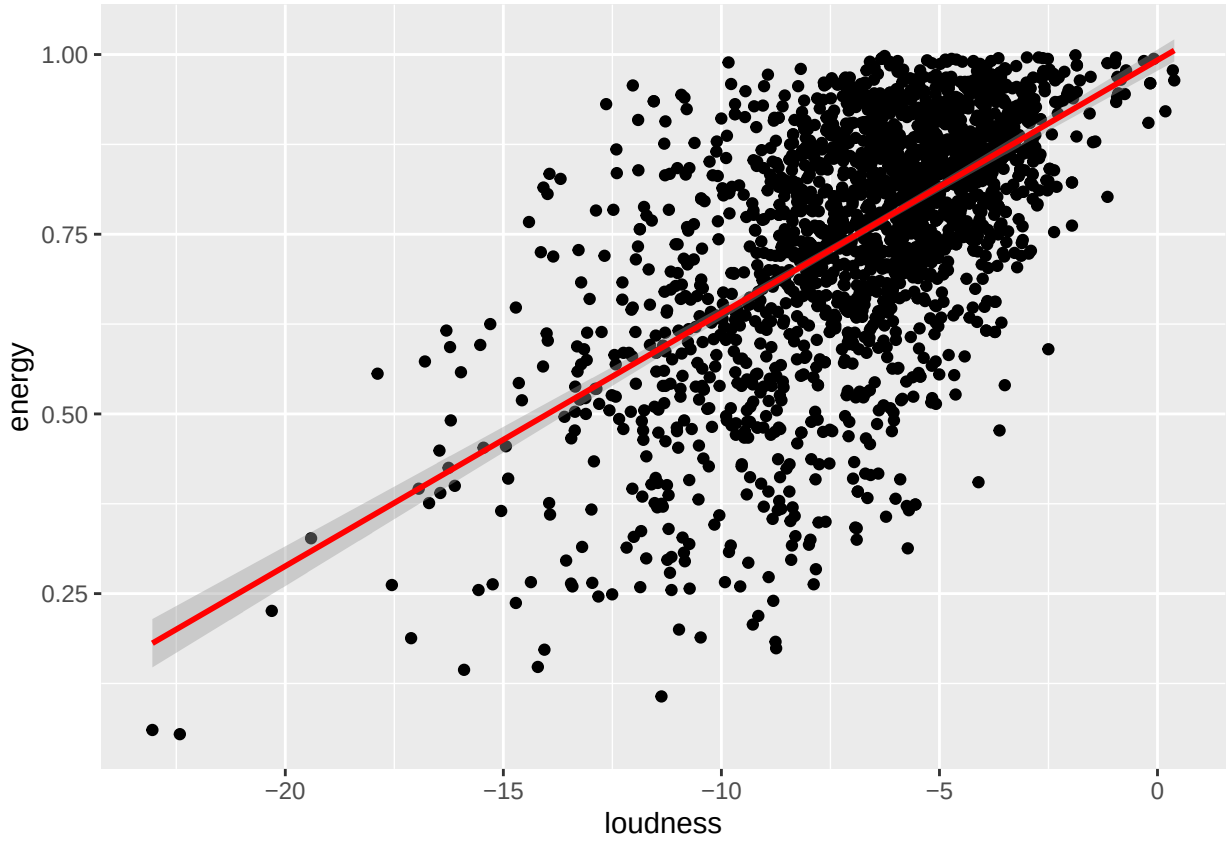

We can observe a possible corruption in the dataset in instrumentalness and acousticness. Is it likely that so many tracks have an instrumentalness or acousticness of 0? Or is this an error within the dataset?

There also appears to be little evidence for a linear ascociation between these audio features.

Based on the context of this problem, I would expect tracks with high energy would also have high loudness. Lets observe this ascociation.

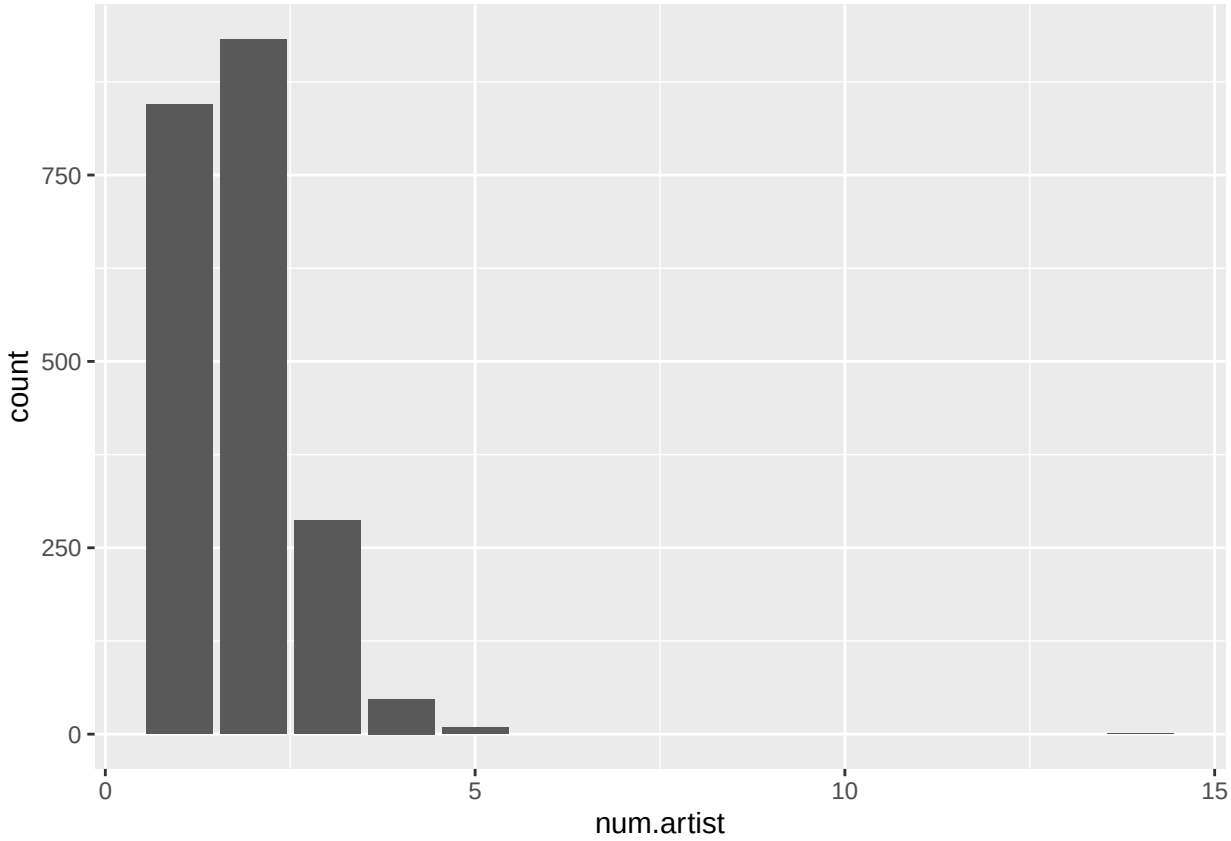
```
louden <- ggplot(data = track_data, aes(x = loudness , y = energy)) + geom_point()+ stat_smooth(method =
louden
```

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




Lets plot the number of artists on a track.

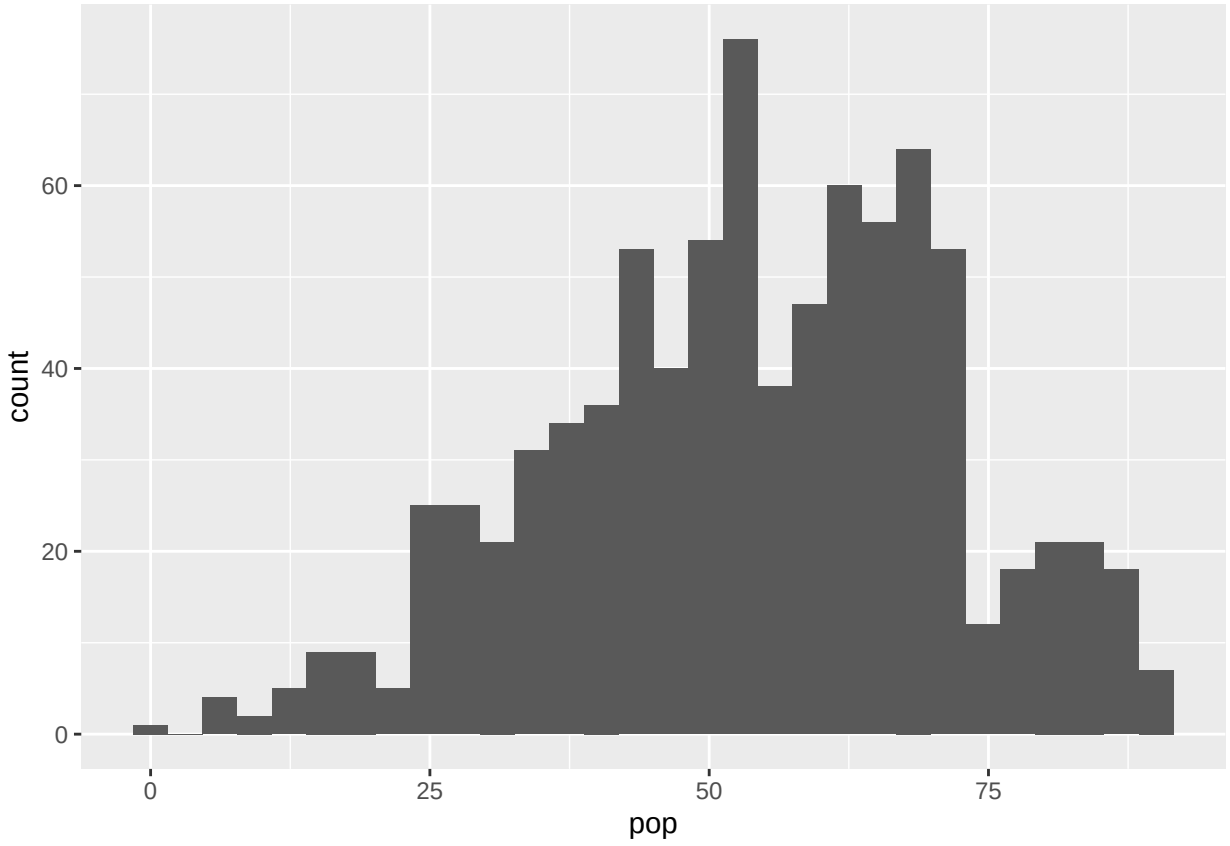
```
t_nart <- ggplot(track_data, aes(x=num.artist)) + geom_bar()
t_nart
```




We can see that the vast majority of number of artists for a track are between [1,3] Lets plot the sum popularity for tracks with number of artists between [1,3]

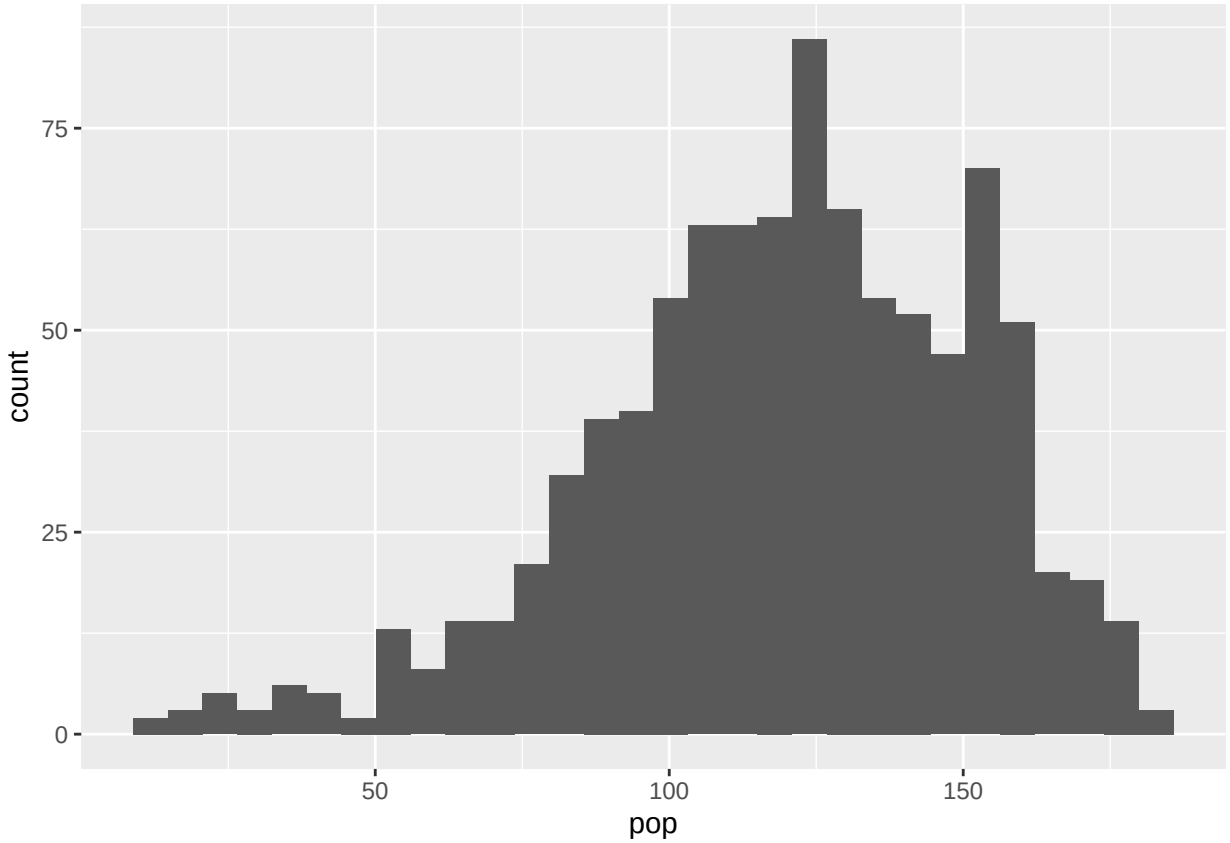
```
data_nart1 <- track_data %>% filter(num.artist == 1)
data_nart2 <- track_data %>% filter(num.artist == 2)
data_nart3 <- track_data %>% filter(num.artist == 3)

p_nart1 <- ggplot(data_nart1, aes(x=pop)) + geom_histogram()
p_nart2 <- ggplot(data_nart2, aes(x=pop)) + geom_histogram()
p_nart3 <- ggplot(data_nart3, aes(x=pop)) + geom_histogram()

p_nart1
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



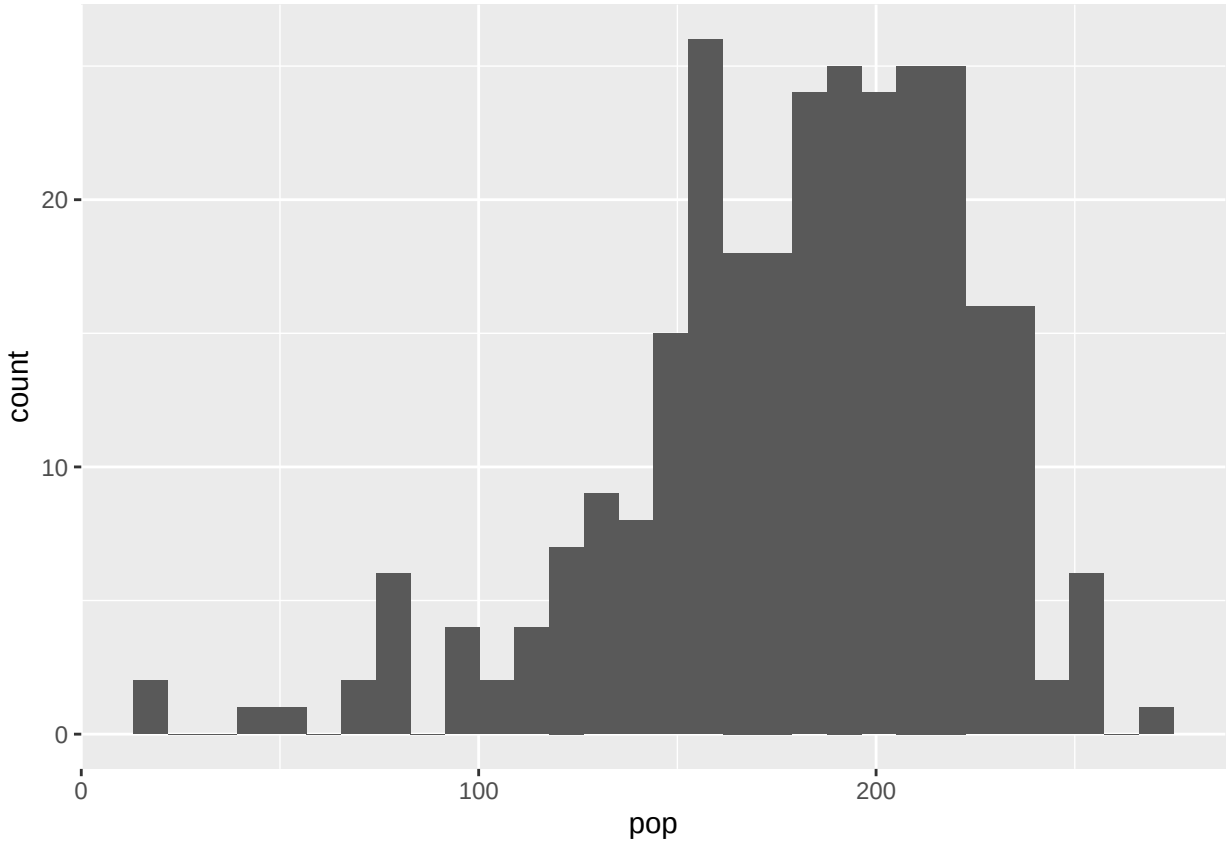

```
p_nart2
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
p_nart3
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


count
20
10
0
0
100
200
pop