Exploratory Analysis of Unsigned Artist Streaming & Sales Data

Sam Roberts-Baca

2021-03-17

Contents

RESEARCH QUESTION

A new, unsigned music artist who has a variety of genres of music released on various streaming services desires to know which of his songs perform the best in terms of streams and sales. What kind of music (in terms of genres / instruments used) should he produce next to generate the most revenue / streams?

In 2020, I released 31 songs of various genres under the stage name "John King Cave". The data associated with these songs serves as the basis of our analysis.

SIGNIFICANCE OF PROJECT

This project will be of significance primarily for unsigned artists like myself who are interested in self-releasing their music online, and want to know how their music performs on various streaming platforms in terms of earnings and stream counts.

Using this approach, unsigned musicians determine which instruments and genres perform the best in terms of streams and sales for their own discographies.

For the general audience, this project will be of significance for those interested in how streaming services convert streams into earnings.

DESCRIPTION OF DATASETS USED

31 songs from the artist John King Cave were released throughout 2020. The following two datasets combine song-specific artist generated data and streaming/sales data obtained from DistroKid, the music distribution service, between Jan. 2020 - Feb. 2021.

Artist-Generated Data

The first dataset used was generated by myself, and stored in "JKC_Data.csv". In total there are 31 songs (observations) of 41 variables represented in this dataset.

The fields specified in "JKC_Data.csv" are as follows:

- Album Title [String] - The name of the album/release the song comes from. For singles, the album title/song title is the same.
- Song Title [String] - The name of the song.
- Track Number [Integer] - The position of the song in the album, starting with 1.
- Lyrics [String] - The lyrics for the song.
- Duration [String] - The duration of the song, in HH:MM:SS format.
- Release Date [String] - The release date of the song, in M/D/YYYY format. Ex: 1/12/2020
- Instruments [String] - A list of instruments played in the song, separated by ';' characters.
- Instrument fields [Boolean] - Each of the following fields indicate whether or not a song uses the given instrument:
  - Synthesizer
  - Pedal Steel
  - Vocals
  - Backup Vocals
  - Banjo
  - Clarinet
  - Alto Sax
  - Tambourine
  - Kneeclaps
  - Harmonica
  - Electric Guitar
  - Bass guitar
  - Keyboard
  - Piano
  - Classical Guitar
  - Trumpet
  - Acoustic Guitar

  - Flute
  - Trombone
  - Singing Saw
  - Drums

- Genres [String] - A list of genres associated with the song, separated by ':' characters.
- Genre fields [Boolean] - Each of the following fields indicate whether or not a song has the given genre:
  - Folk
  - Country
  - Pop
  - Americana
  - Electronic
  - Rock
  - HipHop
  - Blues
  - Alternative
  - Indie
  - Gospel
- ISRC [String] - The International Standard Recording Code (ISRC) associated with the song. ISRCs are a unique identification system for sound recordings and music video recordings. Each ISRC code identifies a specific unique recording.

DistroKid Sales Data

The second dataset used was obtained from DistroKid, a music distribution service. DistroKid is an independent digital music distribution service, which principally offers musicians and other rights-holders the opportunity to distribute and sell or stream their music through online retailers such as iTunes, Spotify, Pandora, Amazon Music, Google Play Music, Tidal, Deezer, iHeartRadio and others.

The DistroKid dataset includes the streaming/sales data associated with each song, saved as "DistroKid_JohnKingCave.csv". In total there are 1084 sales reports (observations) of 10 variables represented in this dataset, for the reporting period between January 2020 to February 2021, inclusive. We will define a sale to mean a specific amount in earnings associated with a specific number of streams for a song in a given sale month for a given country.

The fields specified in "DistroKid_JohnKingCave.csv" are as follows:

- Reporting Date [String] - The date DistroKid reported the sale to the user, in M/D/YYYY format.
- Sale Month [String] - The month the sale took place, in YYYY-MM format.
- Store [String] - The name of the streaming service.
- Artist [String] - The name of the artist. In this set, all observations are labeled "John King Cave".
- Title [String] - The name of the song.
- ISRC [String] - The International Standard Recording Code (ISRC) associated with the song.
- Quantity [Integer] - The number of streams associated with the song for the respective sale. What constitutes a "stream" varies according to each store. In general one stream is determined by some arbitrary length of time listened to a song. For example, in Spotify, approx. 40 seconds listening time = 1 stream
- Song/Album [String] - Specifies whether the release is a song or album. In this set, all observations are labeled "Song".
- Country of Sale [String] - Two-character Country code associated with the given sale.
- Earnings.USD [Numeric] - The amount of money in USD generated for a song for the respective sale.

DATA PREPARATION

To prepare the data for analysis, both datasets were read in R and cleaned. For the artist data, the “Instruments” and “Genres” fields were removed, as the same information was captured in the individual boolean Instrument and Genre fields.

For the DistroKid Sales data, the “Reporting.Date”," Artist“,”Title“, and”Song/Album" fields were removed.

Finally, the Artist-Generated and DistroKid Sales data were merged by the ISRC field. The resulting merged dataset contained 1109 observations of 45 variables.

Although 31 songs were originally used in our input, four were removed during the merge of the two datasets. “Emotion”, “Tread Light”, and “2062 CE” were removed because they were not released on DistroKid. “Just for a Moment” was removed because there was no sales/streaming data available from DistroKid at the time the sales data was downloaded. In total there are 27 songs represented in our analysis.

METHODOLOGY

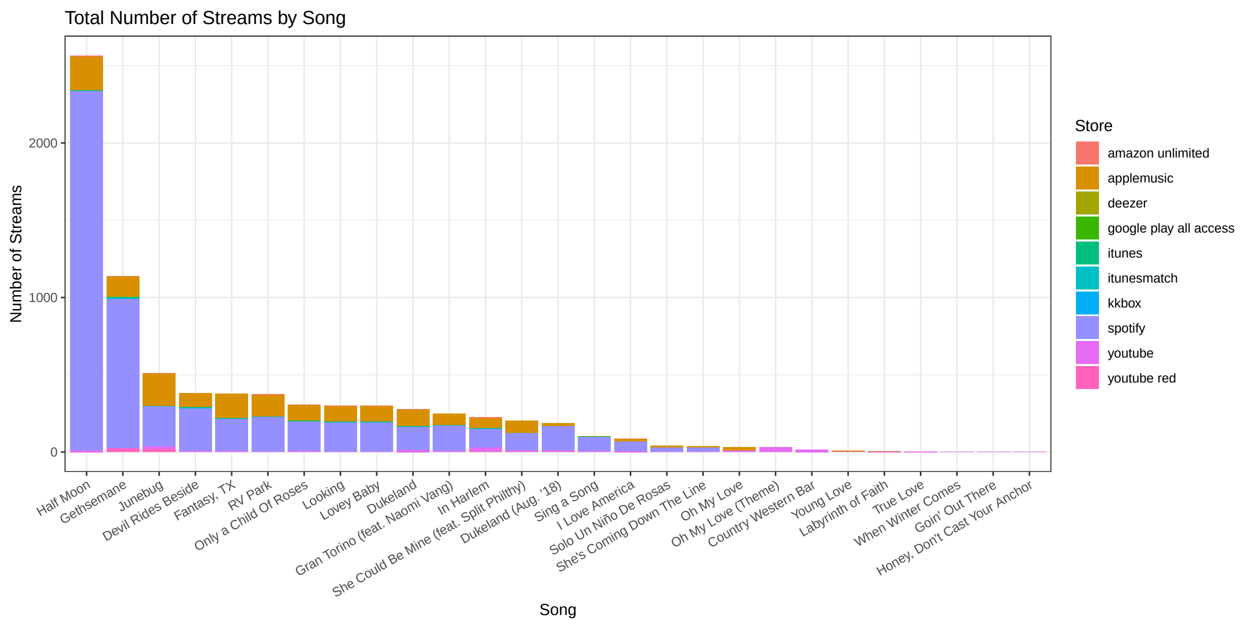
In order to determine what kind of music (in terms of genres / instruments used) an unsigned artist should produce next to generate the most revenue / streams, we will make use of the following methodology:

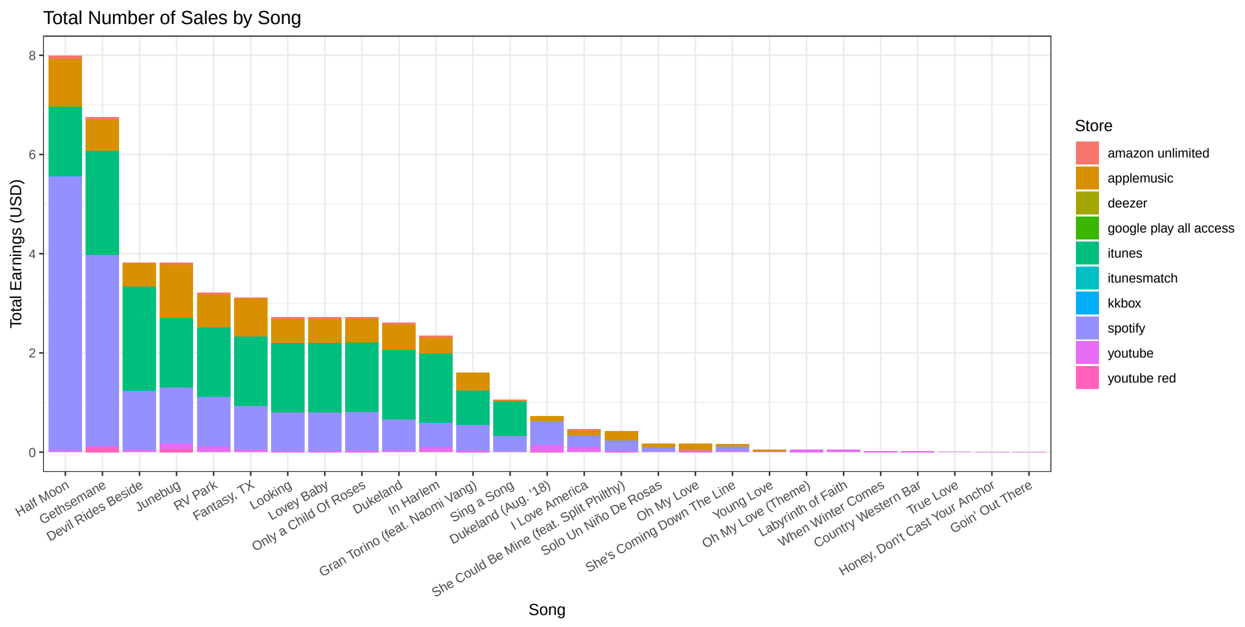
1. Conduct an initial analysis to determine which songs had the most streams/sales on which streaming platforms.
2. Determine the number of sales per stream using a linear regression model.
3. Find out which instruments/genres pair with which songs, and use Principal Component Analysis (PCA) and Factor Analysis (FA) to map the best selling/streaming genres and instruments.
4. Using the results of PCA and FA, conduct a hypothesis test to determine if songs with a given genre/instrument have a higher average number of streams and earnings than the average of the entire discography.

These above steps will be laid out in the following sections.

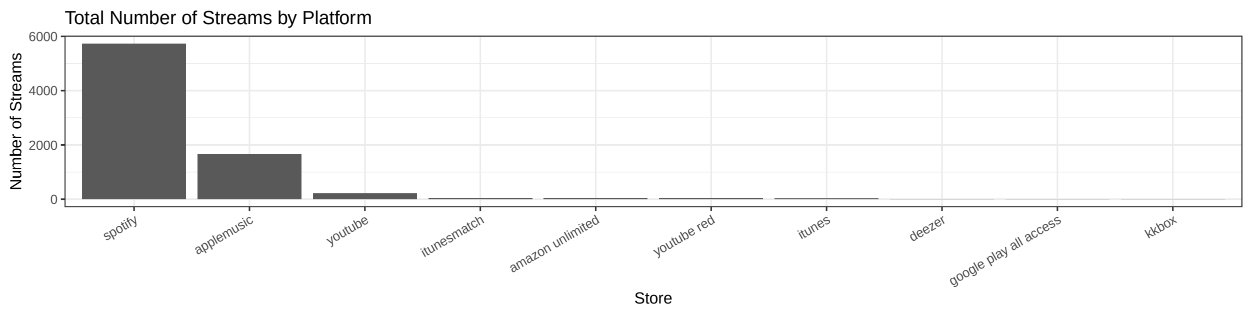
Initial Analysis

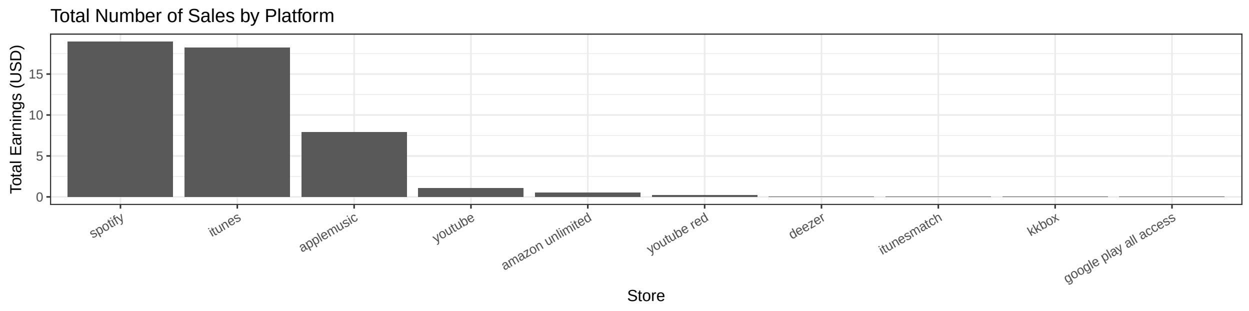
To begin our exploratory analysis, we first look at the general trends. We first transform the data to reflect the total number of streams and sales for each song by store (streaming platform). We can then look at the top songs by number of streams and earnings for each platform:



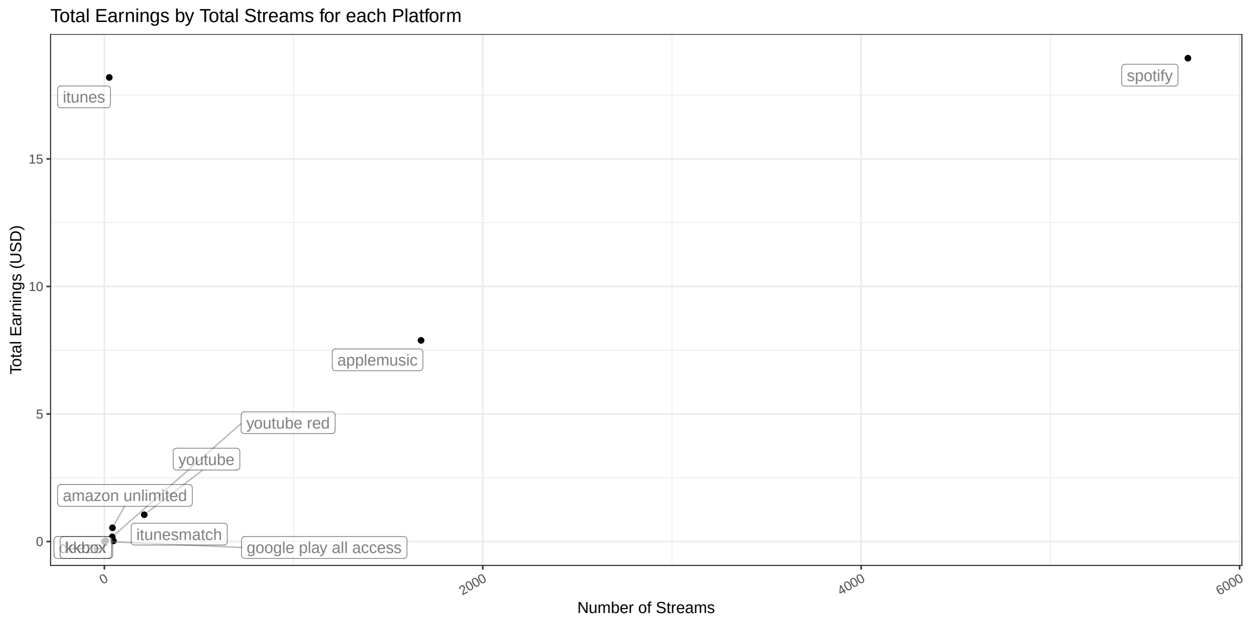


The top two songs released, both in terms of streams and sales, were “Half Moon” and “Gethsemane”. The general trend appears to be that the majority of both streams and earnings are dominated primarily by Spotify, iTunes, and Apple Music. We will confirm this by plotting the total number of streams and earnings by platform:





We can clearly observe that the top two stores for most streams in the given reporting period are Spotify and Apple Music. The top three stores for most sales in the given reporting period are Spotify, iTunes, and Apple Music. All other services are clear outliers in terms of sales and streams. We can also look at at total earnings by total number of streams for each platform:

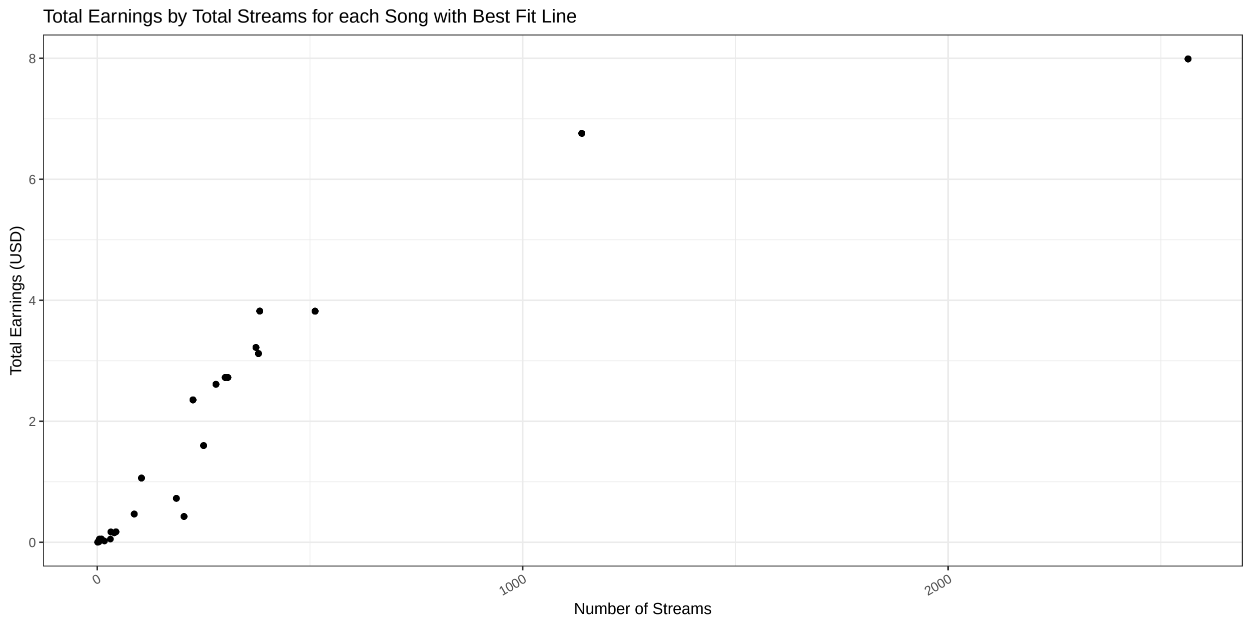


While the amount of earnings by streams varies drastically for each platform, in general there seems to be a relationship between number of streams and total earnings. iTunes is the clear exception when it comes to this trend, which may be attributed to the fact that songs must be purchased directly from the iTunes store in order to be streamed. For this platform, although the number of streams is low, the total earnings is the second highest for any of the given streaming services.

Linear Regression

We will now determine the average amount of money one stream can make for all platforms by using a linear regression best fit line.

We begin by plotting total earnings by total number of streams for each song:



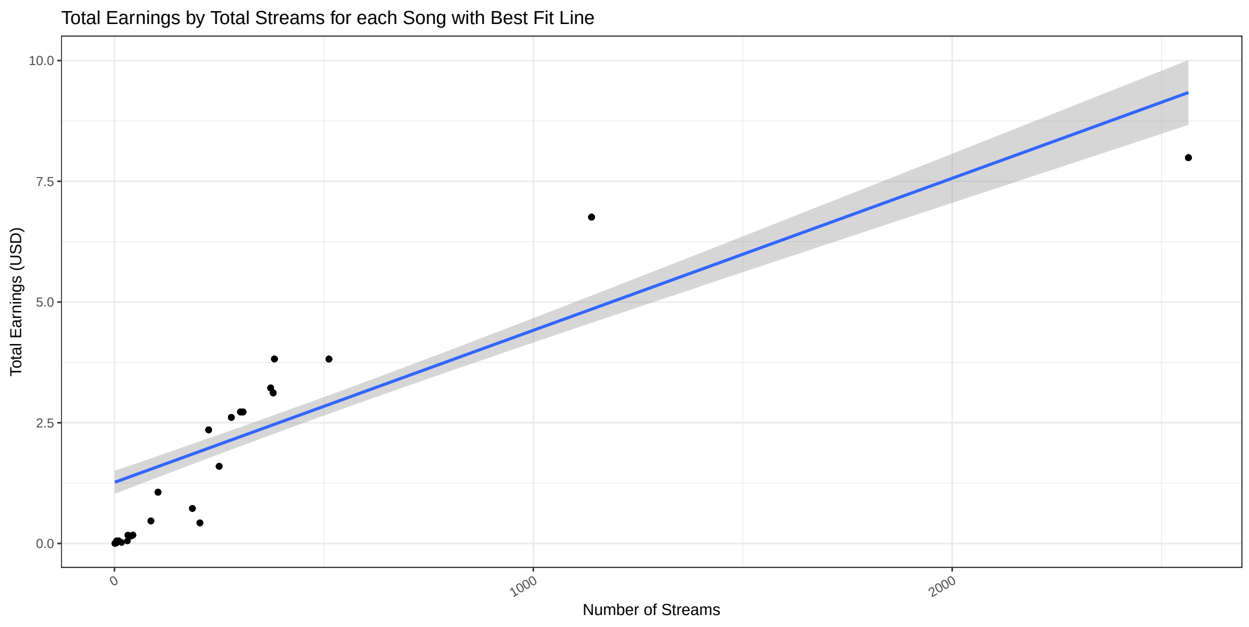
In the above plot, each song corresponds to a single black point. Here we can see that there is a very likely relationship between number of streams and number of earnings. We can confirm this by fitting our data with a linear regression line:

```
Call:
lm(formula = TotalEarnings.USD ~ TotalQuantity, data = dat.song_summary)

Residuals:
    Min      1Q  Median      3Q     Max
-1.4811 -1.1941  0.4678  0.6607  1.9067

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.2651617  0.1203383   10.51   <2e-16 ***
TotalQuantity 0.0031487  0.0001545   20.38   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.047 on 115 degrees of freedom
Multiple R-squared:  0.7831,    Adjusted R-squared:  0.7812
F-statistic: 415.2 on 1 and 115 DF,  p-value: < 2.2e-16
```



Given that our Rˆ2 = .773 and p-value is 7.005e-08 for our linear regression model, we are fairly certain there is a correlation between streams and sales.

According to our linear fit examining earnings over streams by song, for each stream there is an expected increase in approximately .003 US dollars. Therefore, if the unsigned artist wanted to make a million dollars from streams, they would have to have at least 333 million streams.

PCA And Factor Analysis

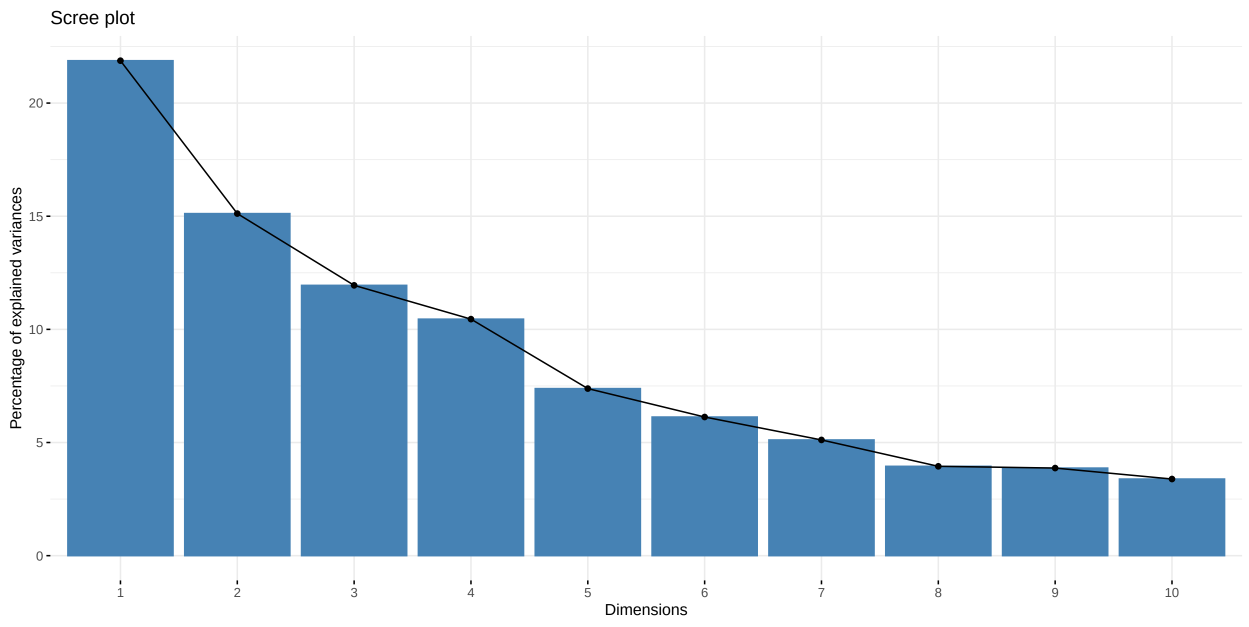
For the next part of our exploratory analysis, we want to examine the qualities associated with the top selling/streaming songs with regard to the instruments and genres employed by each song. We will use Principal Component Analysis (PCA) and Factor Analysis (FA) to accomplish this task.

In order to prepare our data for PCA and FA, we strip the data set down to only the Instrument and Genre columns, as well as the Total Quantity (of streams) and Total Earnings (in USD) for each song. We arrive at a new condensed dataset with 27 observations (songs) with 33 variables.

Principal Component Analysis

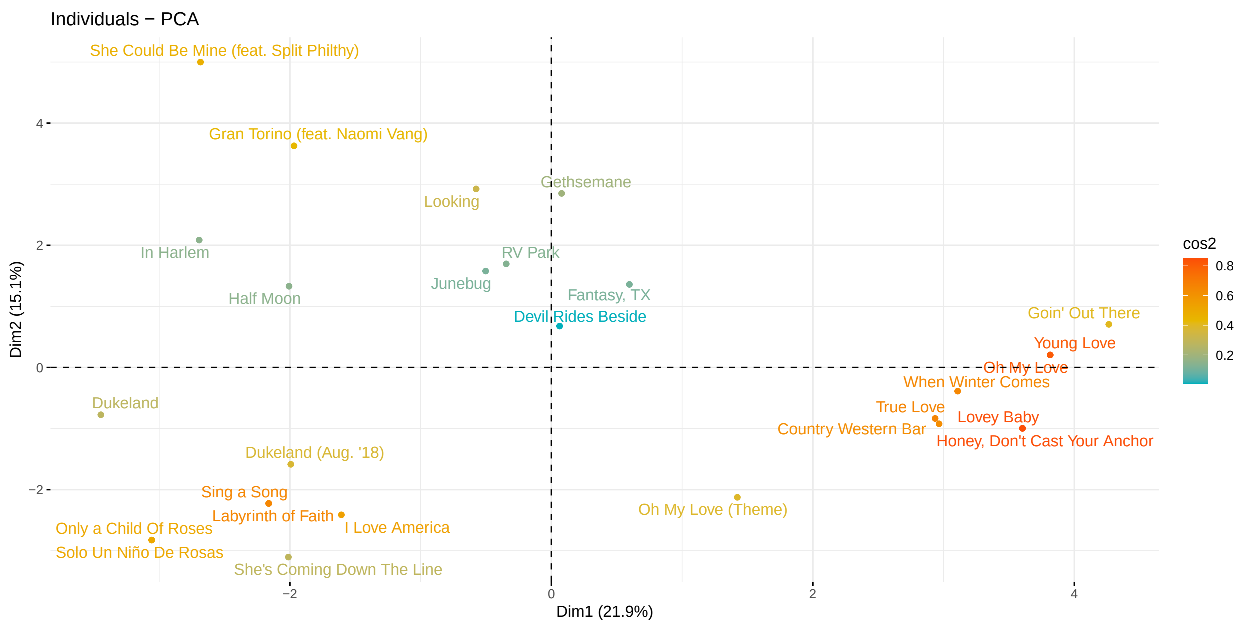
For our PCA, we used a library called "factoextra" in R. In brief, PCA takes a matrix of observations and associated factors and performs a dimensional reduction to turn it into a new correlation matrix by determining its principal components (factors/variables with the most influence on observations). In this case, we will be looking at PCA with a two-dimensional reduction of factors.

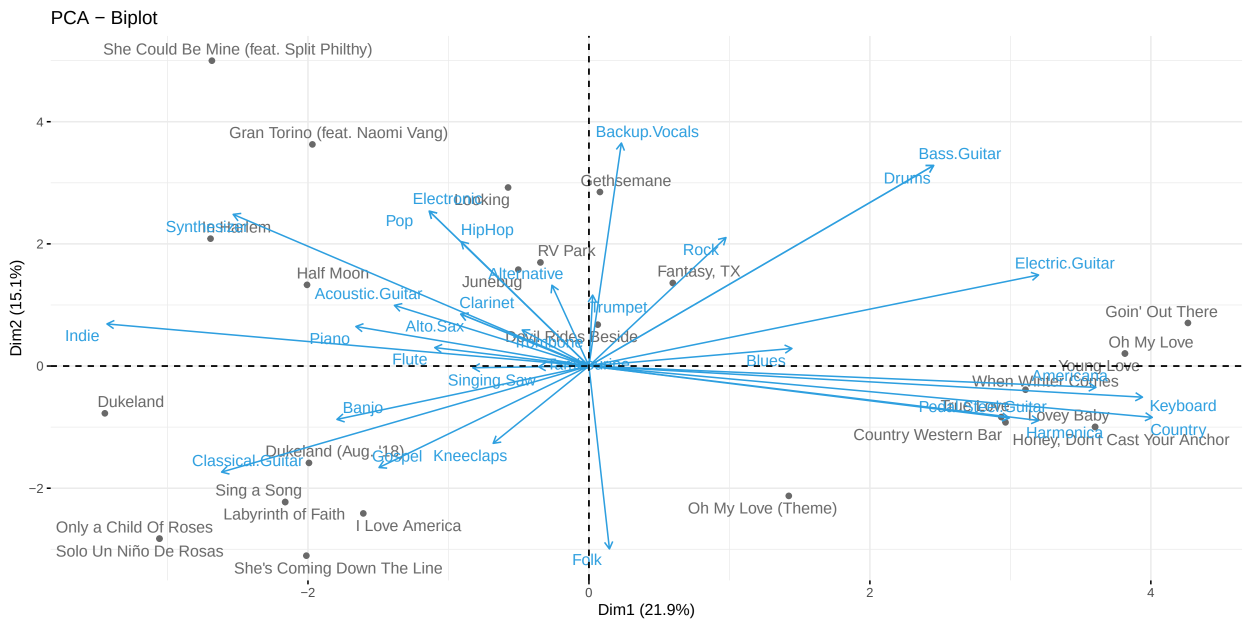
We first construct a scree plot (a line plot of the eigenvalues of factors / principal components in an analysis) to determine the primary dimensions PCA will use. The scree plot for our PCA below shows the percentage of explained variances of our most influential dimensions as determined by our PCA:

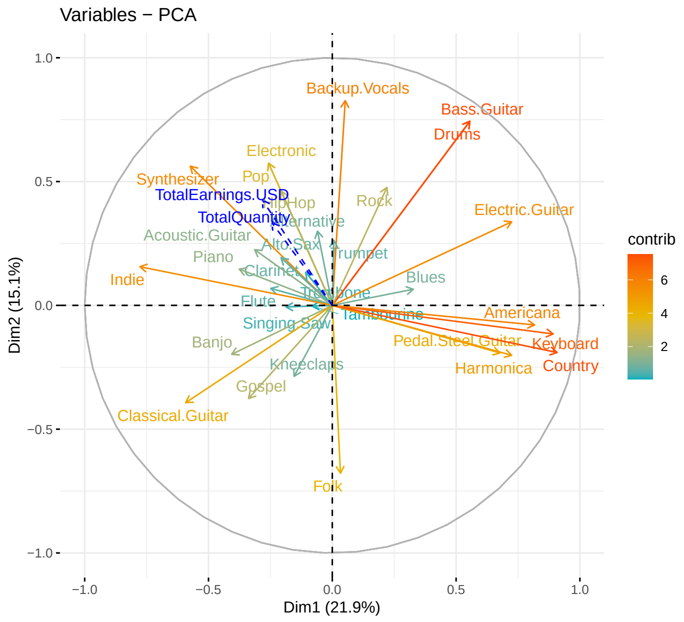


Given the scree plot for our PCA, it is likely that the lower dimensions (at least dimensions 3-6) have a stronger influence on the true correlations between the instruments and genres than what our PCA will show. It will still be helpful to show the results of our PCA in any case, even if we only use the two most influential dimensions.

In order to include the two quantitative fields (TotalQuantity and TotalEarnings.USD) in our analysis, we predict the coordinates of each variable by computing the correlation of each against the other non-quantitative categorical variables in our dataset.







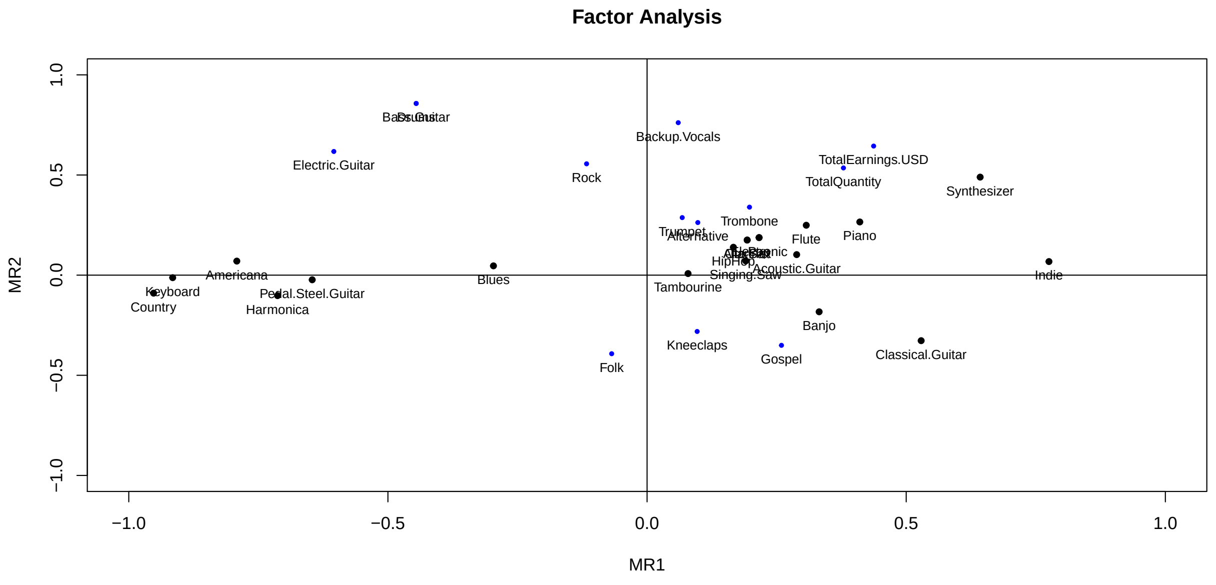
The first plot (Individuals - PCA) shows the correlations associated among individual songs according to the dimensions chosen by the PCA algorithm. There appear to be three main clusters: Country/Americana related songs ("Country Western Bar", "Honey Don't Cast Your Anchor", "When Winter Comes", etc.), Folk songs with only guitar and vocals ("Sing a Song", "Dukeland (Aug. '18)", "Only a Child of Roses", etc.), and Pop/Rock songs with electronic influence ("She Could Be Mine", "Gran Torino", "Looking", etc.).

The second plot (PCA - Biplot) shows the various instruments / genres that correlate with each observation (song) according to the PCA. Arrows between variables with an obtuse angle are said to be negatively correlated, arrows between variables with a right angle have no correlation, and variables with acute angles are said to be positively correlated. According to this plot, the harmonica, pedal steel guitar, and keyboard are positively correlated with the Country/Americana cluster of songs. The classical guitar and Banjo are correlated to the sparse Folk songs, and the Synthesizer correlates fairly well with both Indie and Pop songs.

The third plot (Variables - PCA) shows only the correlations of our variables, now including our two quantitative variables (TotalEarnings.USD and TotalQuantity). Interestingly enough, these two variables appear to correlate most strongly with the Synthesizer instrument, and the Indie, Pop, and Electronic genres. In other words, judging by the results of our PCA, it would appear that the top earning songs and streaming songs are those with synthesizers in the Indie, Pop, and Electronic genres.

Factor Analysis

We will now determine if we achieve similar results using Factor Analysis. Factor Analysis is similar to PCA in that it maps variables in terms of their correlations to one another, but does so by specifying a number of groups (factors) to reduce our variables into. Unlikely PCA, it does not perform a dimensional reduction by breaking down observations by their principal components. We will perform our factor analysis using the fa() function in R using the varimax rotation algorithm.



The blue dots represent the first factor group, and the black dots represent the second factor group. With two factors selected, our Factor Analysis does not appear to group instruments/genres into intuitive groups. However, it does appear to cluster instruments and genres in a very similar way to PCA. Most importantly, the TotalEarnings.USD and TotalQuantity variables are again closest to the Synthesizer instrument. The results from both PCA and Factor Analysis appear to suggest that songs with the Synthesizer are closely related to the songs that earn the most money and streams.

Hypothesis Testing

To confirm whether the inclusion of the synthesizer leads to an increase in total earnings and number of streams across platforms, we will conduct a hypothesis test. Specifically, we want to test the following two hypotheses:

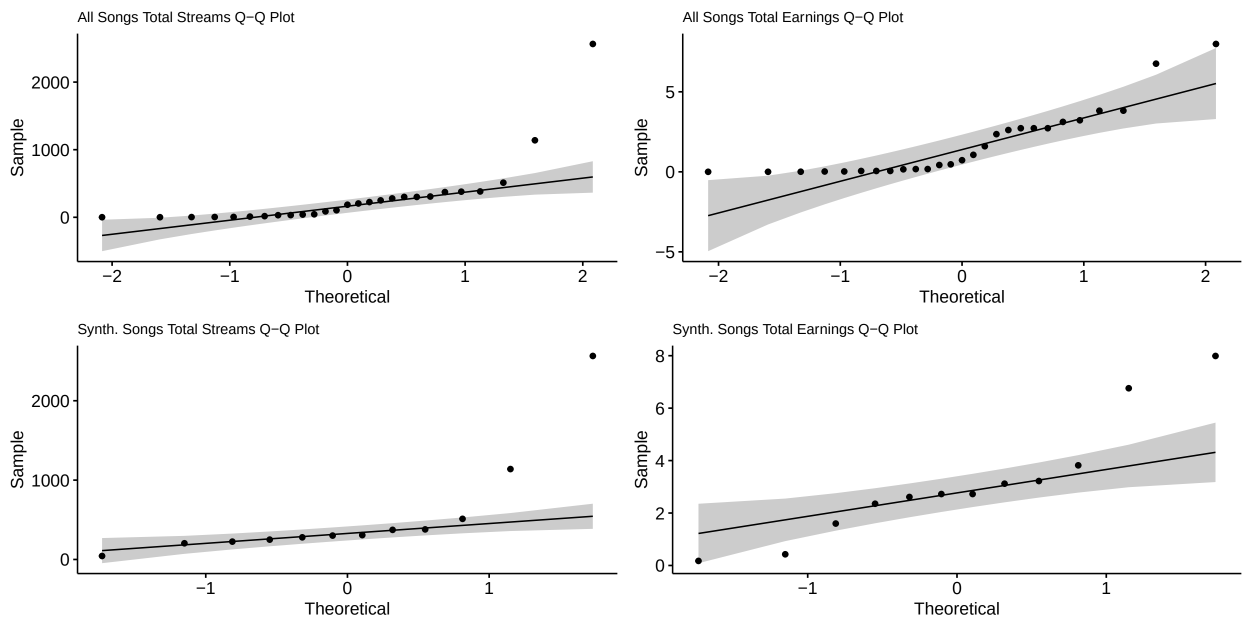
Null Hypothesis 1: The average total number of streams for songs with a synthesizer is equal to the average total number of streams for songs without a synthesizer.

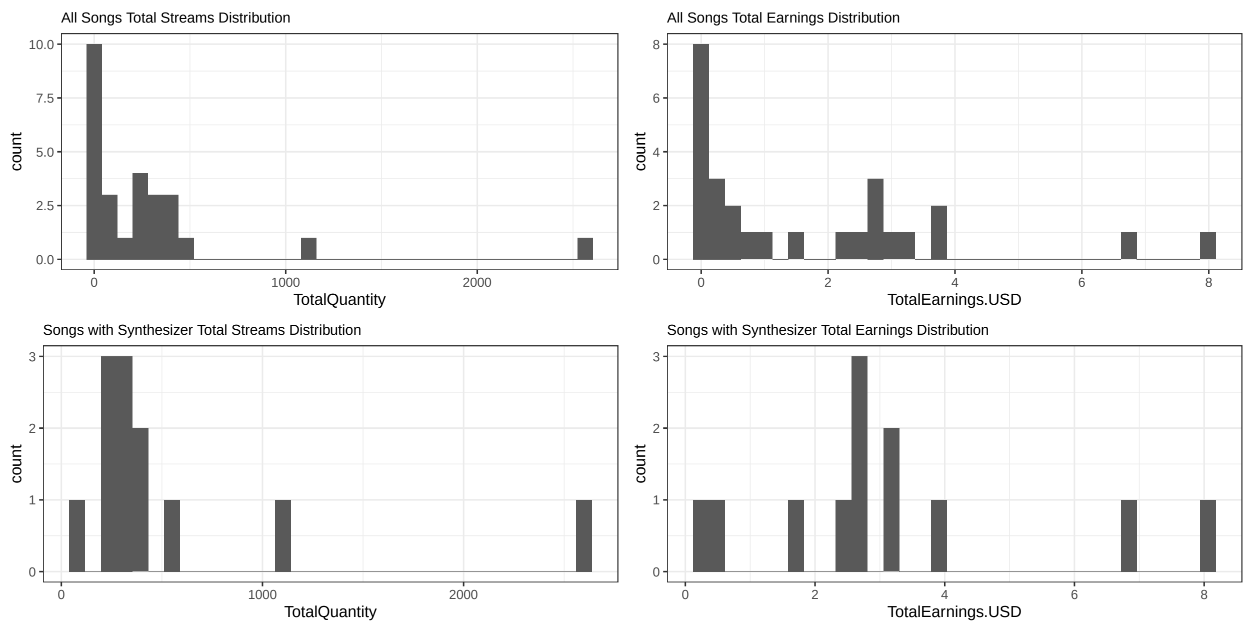
Alternative Hypothesis 1: The average total number of streams for songs with a synthesizer is greater than the average total number of streams for songs without a synthesizer.

Null Hypothesis 2: The average total earnings for songs with a synthesizer is equal to the average total number of earnings for songs without a synthesizer.

Alternative Hypothesis 2: The average total earnings for songs with a synthesizer is greater than the average total number of earnings for songs without a synthesizer.

To test these hypotheses, it occurs to me to use a t-test, as we are dealing with a small sample size (n < 30), and for both tests we are trying to test a sample mean against a population. In order to conduct a t-test however, we must first determine if both the sample and population data are reasonably normally distributed. The below two plots show the Q-Q plots and histograms for the value of TotalQuantity and TotalEarnings.USD for the population (all songs), as well as for the sample (the songs that include a synthesizer instrument):





The above Q-Q plots suggest that our data is reasonably normal for both the sample and population distributions for all songs and just songs with synthesizer, with the exception of a heavy tails, particularly the upper tails for the distributions of TotalQuantity (number of stream) variables.

We will conduct t-tests for both hypotheses, as well as a bootstrapped t-test using a function obtained from https://rdrr.io/cran/MKinfer/src/R/boot.t.test.R with 1000 iterations for greater test accuracy.

The results of our t-tests using $\alpha = 0.05$ are as follows:

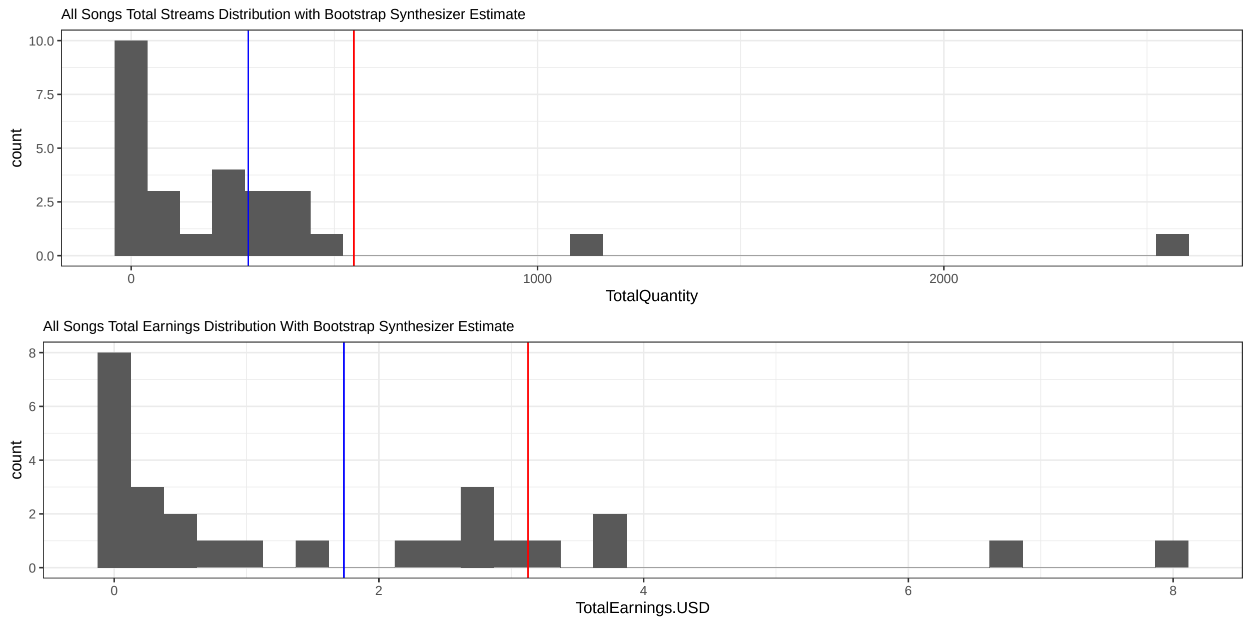
```
    Bootstrapped One Sample t-test

data:  synthesizer_songs$TotalEarnings.USD
bootstrapped p-value = 0.01
95 percent bootstrap percentile confidence interval:
 2.178524      Inf

Results without bootstrap:
t = 2.1194, df = 11, p-value = 0.02882
alternative hypothesis: true mean is greater than 1.735695
95 percent confidence interval:
 1.947996      Inf
sample estimates:
mean of x
  3.12663
```

According to the results of both bootstrap t-tests, our resulting p-value in both cases is less than .05, which means we will reject the null hypothesis in both cases, and conclude that the average total number of earnings and streams for songs with a synthesizer are greater than the average total number of earnings and streams for songs without a synthesizer. It should be noted that our t-tests for Hypothesis 2 returned smaller p-values value for both the bootstrapped and non-bootstrapped cases than did our tests for Hypothesis 1.

Lastly, we will graph the sample mean estimates as returned from our bootstrapped t-tests against the population values:



The red line indicates the average value of the average total number of streams and earnings, respectively, for songs including a synthesizer, overlaid on the population values for both criteria. The blue line indicates the average total number of streams and earnings, respectively, for all of the songs in the population. We can see that the average total number of streams and earnings for songs with the synthesizer is greater than the average total number of streams and earnings for all songs.

CONCLUSION

For John King Cave specifically, it would appear that if the artist wants to make more money and streams off of his music (almost twice as much on average for both metrics), he should consider making more songs with a synthesizer. Pop, Indie, Electronic music with backup vocals and piano accompaniment could potentially be fruitful as well. The most successful streaming platforms for John's music are Spotify, iTunes, and Apple Music. It is also helpful to know that the average number of money per stream for any given platform is 0.003 US Dollars.

For any artist, the technique outlined in this research is an effective strategy for comparing categorical variables to quantitative variables, specifically for determining which instruments and genres sell and stream the best for a given artist. It is also useful to know which stores perform the best for unsigned artists in terms of sales and streams as a whole.

Future opportunities for research include hypothesis testing of other variables against the population (such as Pop, Indie, and Electronic genres), including a larger set of songs to draw from for analysis, and using PCA/FA methods in combination with semantic analysis of lyrical content to determine what influence lyrics have on sales/streaming outputs.

APPENDIX: The Code

```r
knitr::opts_chunk$set(fig.align="center",
                      fig.height=6,
                      fig.width=12,
                      warning = FALSE,
                      message = FALSE,
                      comment = NA,
                      echo=FALSE)
library(tidyverse)
library(knitr) #allows you to create Appendix with all_labels()
library(tidyverse)
library(dplyr) #tibble
library(ISLR) #Auto dataset
theme_set(theme_bw()) #sets default ggplot output style

library(ggpubr)
library(boot)
library(ggrepel)
library(psych)
library(dplyr)

# Reading in / parsing data

# Artist-generated dataset
dat <- read.csv("JKC_Data.csv",stringsAsFactors = FALSE, fileEncoding="UTF-8-BOM",
                 sep=",",header = TRUE)

# Remove redundant Instruments/Genres columns
drops <- c("Instruments","Genres")
dat <- dat[ , !(names(dat) %in% drops)]

# DistroKid Sales dataset
sales_dat <- read.csv("DistroKid_JohnKingCave.csv",stringsAsFactors = FALSE, fileEncoding="UTF-8-BOM",
                 sep=",",header = TRUE)
# Turn blanks/empty spaces into NA
sales_dat[sales_dat==""]<-NA

# Drop unnecessary sales_dat columns
drops <- c("Reporting.Date","Aritst","Title","Song.Album")
sales_dat <- sales_dat[ , !(names(sales_dat) %in% drops)]

# Merge the Artist-Generated and DistroKid Sales data by "ISRC" variable
dat.all <- merge(dat,sales_dat,by="ISRC")


# Initial Analysis

# Total number of Sales/Streams by Song
dat.song_summary <- dat.all %>% group_by(ISRC, Song.Title, Store) %>%
    summarize(Quantity = sum(Quantity),
              Earnings.USD = sum(Earnings.USD))

dat.song_summary_totals <- dat.song_summary %>% group_by(ISRC, Song.Title) %>%
    summarize(TotalQuantity = sum(Quantity),
              TotalEarnings.USD = sum(Earnings.USD))
dat.song_summary_totals_titles <- dat.song_summary_totals

drops <- c("Song.Title")
dat.song_summary_totals <- dat.song_summary_totals[ , !(names(dat.song_summary_totals) %in% drops)]

dat.song_summary <- merge(dat.song_summary,dat.song_summary_totals,by="ISRC")
dat.song_summary <- unique( dat.song_summary )


g1 <- ggplot(dat.song_summary,aes(x= reorder(Song.Title,-TotalQuantity),Quantity, fill=Store))+geom_bar
g1

g2 <- ggplot(dat.song_summary,aes(x= reorder(Song.Title,-TotalEarnings.USD),Earnings.USD, fill=Store))+
g2


# Total numbers of Streams/Sales by Platform

store_count <- aggregate(cbind(Quantity = dat.all$Quantity,Earnings.USD = dat.all$Earnings.USD), by=lis

g1 <- ggplot(store_count,aes(x= reorder(Store,-Quantity),Quantity))+geom_bar(stat ="identity") + theme(

g2 <- ggplot(store_count,aes(x= reorder(Store,-Earnings.USD),Earnings.USD))+geom_bar(stat ="identity")
```

```
ggarrange(g1,g2,
          ncol = 1, nrow = 2)


# Total Earnings by Total Number of Streams for each Platform
g3 <- ggplot(store_count,aes(x= Quantity,y = Earnings.USD))+geom_point() + theme(axis.text.x = element_
                  box.padding   = 0.35,
                  point.padding = 0.5,
                  segment.color = 'grey50',
                  max.overlaps = 14,
                  alpha = .5)
g3


# Total Earnings by Total Number of Streams for each Song
g3 <- ggplot(dat.song_summary,aes(x= TotalQuantity,y = TotalEarnings.USD))+geom_point() + theme(axis.te
g3


# Linear Regression Best Fit Line
linear_fit <- lm(TotalEarnings.USD~TotalQuantity,data=dat.song_summary)
summary(linear_fit)

g3 <- g3 + stat_smooth(method = "lm", formula = y ~ x)
g3


# Principal Component Analysis

# Data Preparation
dat.cat <- dat.song_summary_totals_titles[, c("ISRC", "TotalQuantity", "TotalEarnings.USD")]
dat.cat <- unique( dat.cat )
dat.cat <- merge(dat.cat,dat,by="ISRC")

# Drop unnecessary sales_dat columns
drops <- c("ISRC", "Album.Title","Track","Lyrics","Release.Date","Instruments", "Genres", "Sale.Month",
dat.cat <- dat.cat[ , !(names(dat.cat) %in% drops)]
dat.cat$rownames = dat.cat$Song.Title

# Set row indices to be song titles
rownames(dat.cat) <- dat.cat$Song.Title
drops <- c("Song.Title")
dat.cat <- dat.cat[ , !(names(dat.cat) %in% drops)]
dat.cat <- within(dat.cat, rm("rownames"))

# PCA Data Object Creation
library(factoextra)
dat.pca <- prcomp(dat.cat[3:33], scale = TRUE)
fviz_eig(dat.pca)

#summary(dat.pca)
```

```r
# PCA Individuals Plot
fviz_pca_ind(dat.pca,
             col.ind = "cos2", # Color by the quality of representation
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE     # Avoid text overlapping
             )

# PCA BiPlot
fviz_pca_biplot(dat.pca, repel = TRUE,
                col.var = "#2E9FDF", # Variables color
                col.ind = "#696969"  # Individuals color
                )

# Predict coordinates of quantitative variables and compute cos2
quanti.coord <- cor(dat.cat[1:2], dat.pca$x)
quanti.cos2 <- quanti.coord^2

# Graph of variables including supplementary variables
p <- fviz_pca_var(dat.pca,col.var = "contrib", # Color by contributions to the PC
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), max.overlaps = 15, repel = TRUE)

fviz_add(p, quanti.coord, color ="blue", geom="arrow")

# Eigenvalues
eig.val <- get_eigenvalue(dat.pca)
#eig.val

# Results for Variables
res.var <- get_pca_var(dat.pca)
#res.var$coord          # Coordinates
#res.var$contrib        # Contributions to the PCs
#res.var$cos2           # Quality of representation
# Results for individuals
res.ind <- get_pca_ind(dat.pca)
#res.ind$coord          # Coordinates
#res.ind$contrib        # Contributions to the PCs
#res.ind$cos2           # Quality of representation


# Let's see if we obtain similar results with Factor Analysis.
# Factor Analysis

dat.cat <- na.omit(dat.cat)     #remove the cases with missing values
f2 <- fa(dat.cat,2,rotation="varimax")          #factor analyze the resulting item
#(f2)                                             #show the result
load=loadings(f2)
#print(load,sort=TRUE,digits=2,cutoff=0.01)  #show the loadings
#identify(load,labels=names(dat.cat))

#plot factor 1 by factor 2
plot(f2,labels=names(dat.cat), cex=0.75, xlim=c(-1,1), ylim=c(-1,1))
```

```
# Bootstrap t-test function from https://rdrr.io/cran/MKinfer/src/R/boot.t.test.R

boot.t.test <- function(x, ...){
  UseMethod("boot.t.test")
}
boot.t.test.default <- function(x, y = NULL, alternative = c("two.sided", "less", "greater"),
                        mu = 0, paired = FALSE, var.equal = FALSE,
                        conf.level = 0.95, R = 9999, symmetric = FALSE, ...){
  alternative <- match.arg(alternative)
  if(!missing(mu) && (length(mu) != 1 || is.na(mu)))
    stop("'mu' must be a single number")
  if(!missing(conf.level) && (length(conf.level) != 1 || !is.finite(conf.level) ||
                               conf.level < 0 || conf.level > 1))
    stop("'conf.level' must be a single number between 0 and 1")
  if(!is.null(y)){
    dname <- paste(deparse(substitute(x)), "and", deparse(substitute(y)))
    if (paired)
      xok <- yok <- complete.cases(x, y)
    else{
      yok <- !is.na(y)
      xok <- !is.na(x)
    }
    y <- y[yok]
  }else{
    dname <- deparse(substitute(x))
    if (paired)
      stop("'y' is missing for paired test")
    xok <- !is.na(x)
    yok <- NULL
  }
  x <- x[xok]
  if(paired){
    x <- x - y
    y <- NULL
  }
  nx <- length(x)
  mx <- mean(x)
  vx <- var(x)
  if (is.null(y)) {
    if (nx < 2)
      stop("not enough 'x' observations")
    df <- nx - 1
    stderr <- sqrt(vx/nx)
    if (stderr < 10 * .Machine$double.eps * abs(mx))
      stop("data are essentially constant")
    tstat <- (mx - mu)/stderr
    method <- if (paired) "Bootstrapped Paired t-test" else "Bootstrapped One Sample t-test"
    estimate <- setNames(mx, if (paired) "mean of the differences" else "mean of x")
    x.cent <- x - mx
    X <- matrix(sample(x.cent, size = nx*R, replace = TRUE), nrow = R)
    MX <- rowMeans(X)
    VX <- rowSums((X-MX)^2)/(nx-1)
    STDERR <- sqrt(VX/nx)
    TSTAT <- MX/STDERR
    EFF <- MX+mx
  }else{
    ny <- length(y)
    if(nx < 1 || (!var.equal && nx < 2))
      stop("not enough 'x' observations")
    if(ny < 1 || (!var.equal && ny < 2))
      stop("not enough 'y' observations")
    if(var.equal && nx + ny < 3)
      stop("not enough observations")
    my <- mean(y)
    vy <- var(y)
    method <- paste("Bootstrapped", paste(if (!var.equal) "Welch", "Two Sample t-test"))
    estimate <- c(mx, my)
    names(estimate) <- c("mean of x", "mean of y")
    if(var.equal){
      df <- nx + ny - 2
      v <- 0
      if (nx > 1)
        v <- v + (nx - 1) * vx
      if (ny > 1)
        v <- v + (ny - 1) * vy
      v <- v/df
      stderr <- sqrt(v * (1/nx + 1/ny))
      z <- c(x, y)
      Z <- matrix(sample(z, size = (nx+ny)*R, replace = TRUE), nrow = R)
      X <- Z[,1:nx]
      Y <- Z[,(nx+1):(nx+ny)]
      MX <- rowMeans(X)
      MY <- rowMeans(Y)
      V <- (rowSums((X-MX)^2) + rowSums((Y-MY)^2))/df
      STDERR <- sqrt(V*(1/nx + 1/ny))
      EFF <- (MX+mx) - (MY+my)
    }else{
      stderrx <- sqrt(vx/nx)
      stderry <- sqrt(vy/ny)
      stderr <- sqrt(stderrx^2 + stderry^2)
      df <- stderr^4/(stderrx^4/(nx - 1) + stderry^4/(ny - 1))
      z <- c(x, y)
      mz <- mean(z)
      x.cent <- x - mx + mz
      y.cent <- y - my + mz
      X <- matrix(sample(x.cent, size = nx*R, replace = TRUE), nrow = R)
      Y <- matrix(sample(y.cent, size = ny*R, replace = TRUE), nrow = R)
      MX <- rowMeans(X)
      MY <- rowMeans(Y)
      VX <- rowSums((X-MX)^2)/(nx-1)
      VY <- rowSums((Y-MY)^2)/(ny-1)
      STDERR <- sqrt(VX/nx + VY/ny)
      EFF <- (MX+mx) - (MY+my)
    }
    if (stderr < 10 * .Machine$double.eps * max(abs(mx), abs(my)))
      stop("data are essentially constant")
```

```r
    tstat <- (mx - my - mu)/stderr
    TSTAT <- (MX - MY)/STDERR
  }
  if (alternative == "less") {
    pval <- pt(tstat, df)
    boot.pval <- mean(TSTAT < tstat)
    cint <- c(-Inf, tstat + qt(conf.level, df))
    boot.cint <- c(-Inf, quantile(EFF, conf.level))
  }else if(alternative == "greater") {
    boot.pval <- mean(TSTAT > tstat)
    pval <- pt(tstat, df, lower.tail = FALSE)
    cint <- c(tstat - qt(conf.level, df), Inf)
    boot.cint <- c(quantile(EFF, 1-conf.level), Inf)
  }else{
    pval <- 2 * pt(-abs(tstat), df)
    if(symmetric)
      boot.pval <- mean(abs(TSTAT) > abs(tstat))
    else
      boot.pval <- 2*min(mean(TSTAT <= tstat), mean(TSTAT > tstat))
    alpha <- 1 - conf.level
    cint <- qt(1 - alpha/2, df)
    cint <- tstat + c(-cint, cint)
    boot.cint <- quantile(EFF, c(alpha/2, 1-alpha/2))
  }
  cint <- mu + cint * stderr
  names(tstat) <- "t"
  names(df) <- "df"
  names(mu) <- if (paired || !is.null(y)) "difference in means" else "mean"
  attr(cint, "conf.level") <- conf.level
  attr(boot.cint, "conf.level") <- conf.level
  rval <- list(statistic = tstat, parameter = df, p.value = pval,
               boot.p.value = boot.pval,
               conf.int = cint, boot.conf.int = boot.cint,
               estimate = estimate, null.value = mu,
               stderr = stderr, alternative = alternative, method = method,
               data.name = dname)
  class(rval) <- c("boot.htest", "htest")
  rval
}
boot.t.test.formula <- function (formula, data, subset, na.action, ...){
  if (missing(formula) || (length(formula) != 3L) || (length(attr(terms(formula[-2L]),
                                                      "term.labels")) != 1L))
    stop("'formula' missing or incorrect")
  m <- match.call(expand.dots = FALSE)
  if (is.matrix(eval(m$data, parent.frame())))
    m$data <- as.data.frame(data)
  m[[1L]] <- quote(stats::model.frame)
  m$... <- NULL
  mf <- eval(m, parent.frame())
  DNAME <- paste(names(mf), collapse = " by ")
  names(mf) <- NULL
  response <- attr(attr(mf, "terms"), "response")
  g <- factor(mf[[-response]])
  if (nlevels(g) != 2L)
    stop("grouping factor must have exactly 2 levels")
  DATA <- setNames(split(mf[[response]], g), c("x", "y"))
  y <- do.call("boot.t.test", c(DATA, list(...)))
  y$data.name <- DNAME
  if (length(y$estimate) == 2L)
    names(y$estimate) <- paste("mean in group", levels(g))
  y
}
print.boot.htest <- function (x, digits = getOption("digits"), prefix = "\t", ...) {
  cat("\n")
  cat(strwrap(x$method, prefix = prefix), sep = "\n")
  cat("\n")
  cat("data:  ", x$data.name, "\n", sep = "")
  out <- character()
  if (!is.null(x$boot.p.value)) {
    bfp <- format.pval(x$boot.p.value, digits = max(1L, digits - 3L))
    cat("bootstrapped p-value",
        if (substr(bfp, 1L, 1L) == "<") bfp else paste("=", bfp), "\n")
  }
  if (!is.null(x$conf.int)) {
    cat(format(100 * attr(x$boot.conf.int, "conf.level")),
        " percent bootstrap percentile confidence interval:\n",
        " ", paste(format(x$boot.conf.int[1:2], digits = digits),
                   collapse = " "), "\n", sep = "")
  }
  cat("\nResults without bootstrap:\n")
  if (!is.null(x$statistic))
    out <- c(out, paste(names(x$statistic), "=", format(x$statistic,
                                                         digits = max(1L, digits - 2L))))
  if (!is.null(x$parameter))
    out <- c(out, paste(names(x$parameter), "=", format(x$parameter,
                                                         digits = max(1L, digits - 2L))))
  if (!is.null(x$p.value)) {
    fp <- format.pval(x$p.value, digits = max(1L, digits -
                                                3L))
    out <- c(out, paste("p-value",
                        if (substr(fp, 1L, 1L) == "<") fp else paste("=", fp)))
  }
  cat(strwrap(paste(out, collapse = ", ")), sep = "\n")
  if (!is.null(x$alternative)) {
    cat("alternative hypothesis: ")
    if (!is.null(x$null.value)) {
      if (length(x$null.value) == 1L) {
        alt.char <- switch(x$alternative, two.sided = "not equal to",
                           less = "less than", greater = "greater than")
        cat("true ", names(x$null.value), " is ", alt.char,
            " ", x$null.value, "\n", sep = "")
      }
      else {
        cat(x$alternative, "\nnull values:\n", sep = "")
        print(x$null.value, digits = digits, ...)
      }
    }
    else cat(x$alternative, "\n", sep = "")
  }
  if (!is.null(x$conf.int)) {
    cat(format(100 * attr(x$conf.int, "conf.level")), " percent confidence interval:\n",
        " ", paste(format(x$conf.int[1:2], digits = digits),
                   collapse = " "), "\n", sep = "")
  }
  if (!is.null(x$estimate)) {
    cat("sample estimates:\n")
    print(x$estimate, digits = digits, ...)
  }
  cat("\n")
  invisible(x)
}

# Hypothesis Testing

synthesizer_songs <- dat.cat[which(dat.cat$Synthesizer == 1), ,]
synthesizer.quantity <- synthesizer_songs$TotalQuantity
synthesizer.earnings <- synthesizer_songs$TotalEarnings.USD

mu_TotalQuantity <- mean(dat.cat$TotalQuantity)
mu_TotalEarnings.USD <- mean(dat.cat$TotalEarnings.USD)

# Q-Q Plots

qq1 <- ggqqplot(dat.cat$TotalQuantity,title="All Songs Total Streams Q-Q Plot") + theme(plot.title = ele
qq2 <- ggqqplot(dat.cat$TotalEarnings.USD,title="All Songs Total Earnings Q-Q Plot") + theme(plot.title
qq3 <- ggqqplot(synthesizer_songs$TotalQuantity,title="Synth. Songs Total Streams Q-Q Plot") + theme(pl
qq4 <- ggqqplot(synthesizer_songs$TotalEarnings.USD,title="Synth. Songs Total Earnings Q-Q Plot") + the

ggarrange(qq1,qq2,qq3,qq4,
          ncol = 2, nrow = 2)

# Histogram Distribution Testing

h1 <- ggplot(data=dat.cat,aes(x=TotalQuantity)) + geom_histogram(bins=33) + ggtitle("All Songs Total St
h2 <- ggplot(data=dat.cat,aes(x=TotalEarnings.USD)) + geom_histogram(bins=33) + ggtitle("All Songs Tota
h3 <- ggplot(data=synthesizer_songs,aes(x=TotalQuantity)) + geom_histogram(bins=33) + ggtitle("Songs wi
h4 <- ggplot(data=synthesizer_songs,aes(x=TotalEarnings.USD)) + geom_histogram(bins=33) + ggtitle("Song

ggarrange(h1,h2,h3,h4, ncol = 2, nrow = 2)


# Bootstrapped T-tests

#t.test(synthesizer_songs$TotalQuantity, mu = mu_TotalQuantity, alternative = "greater")
boot.test1 <- boot.t.test(x = synthesizer_songs$TotalQuantity, y = NULL,
       alternative = "greater",
       mu = mu_TotalQuantity, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, R = 1000)
```

```
#t.test(synthesizer_songs$TotalEarnings.USD, mu = mu_TotalEarnings.USD, alternative = "greater")
boot.test2 <- boot.t.test(x = synthesizer_songs$TotalEarnings.USD, y = NULL,
       alternative = "greater",
       mu = mu_TotalEarnings.USD, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, R = 1000)
boot.test2


# Graphing Estimated sample means of Bootstrap T-test against population distributions

g1 <- ggplot(data=dat.cat,aes(x=TotalQuantity)) + geom_histogram(bins=33) + ggtitle("All Songs Total St
g2 <- ggplot(data=dat.cat,aes(x=TotalEarnings.USD)) + geom_histogram(bins=33) + ggtitle("All Songs Tota

ggarrange(g1,g2,
          ncol = 1, nrow = 2)
```