A (Time Series) Attack on Bangtan

(3)

Sydney McSoley
Professor Trichtinger
STAT 391
09 May 2023

STAT 391: A (Time Series) Attack on Bangtan

09 May 2023

Exploratory Data Analysis

Time Series

The intervals are every 6 months from their debut in June 2013 until December 2021. The data starts from 2013-06 and ends at 2021-12. The amount of songs released changes with each half-year.

Introduction and Description of Data

BTS is a Korean boy group from HYBE Entertainment. Consisting of RM, Jin, SUGA, J-Hope, Jimin, V and Jungkook, they are one of the most recognizable Korean acts. They are most known for their hit songs "Dynamite", "Butter", "Boy with Luv (ft. Halsey)", and "Fake Love".

1:

The dataset consists of 147 songs by BTS, which includes remixes, solo songs, original Japanese songs and Japanese versions of hit songs. This dataset excludes repeated songs on compilation albums and doesn't include any songs off of "Proof". I broke songs up into groups based off of release time of year. and took the average of each variable.

2: Frequency/Span:

The data spans from June 2013 until December 2021. The intervals are every 6 months (January-June and July-December). The number of songs released in each 6 month period changes. For example, in the first half of 2014, BTS only released "Boy in Luv" while in the second half of 2018, BTS released 21 songs.

3: Cleaning:

With this dataset, I found out how many songs were released in each 6 month period from June 2013 until December 2021. I confirmed that there were no repeats of songs in my dataset.

4: Citations:

Mahajan, Komal & Barman, Dikshita. Spotify... Music for every mood!. Retrieved 06 April, 2023 from https://rstudio-pubs-static.s3.amazonaws.com/605258_4f5e240a79a3405295b1e273de4c29fd.html.

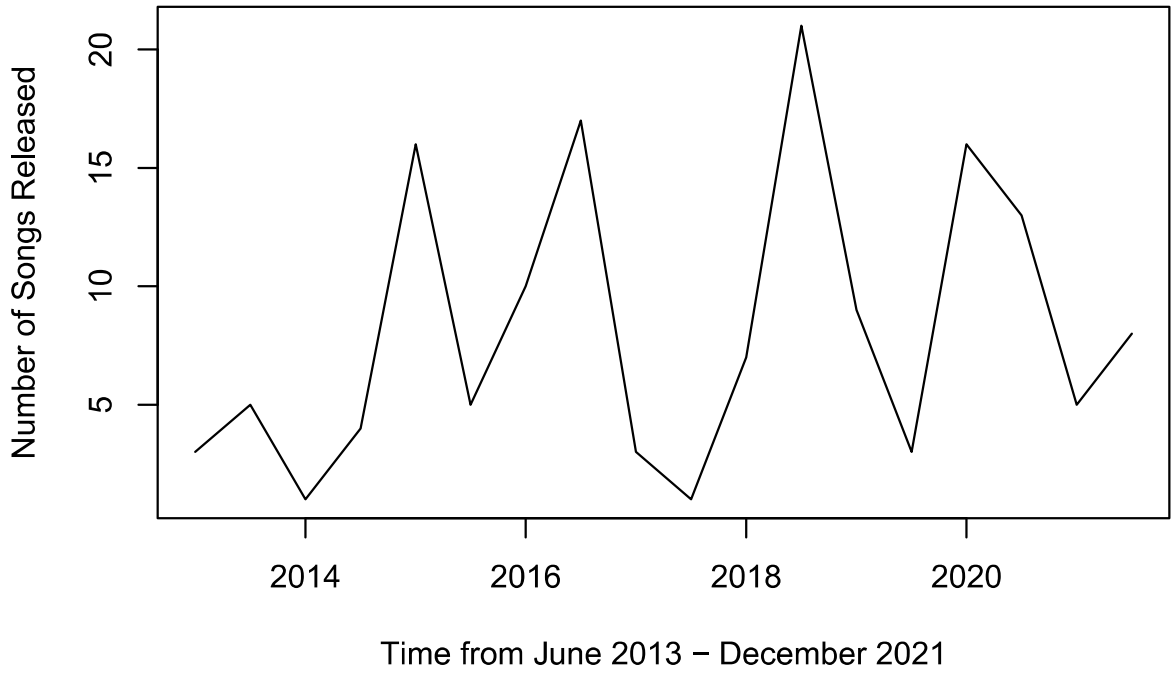
Mogura 2.0. (2021, November). BTS 147 Songs Audio Features (Spotify), Version 1. Retrieved 02 April, 2023 from https://www.kaggle.com/datasets/mogura20/bts-147-songs-audio-features-spotify-api.

Widjojo, Conchita. (2022, April). BTS Shines in Custom Louis Vuitton Suits at the 2022 Grammy Awards. Retrieved 09 May, 2023 from https://wwd.com/fashion-news/fashion-scoops/bts-custom-louis-vuitton-suits-2022-grammy-awards-1235149143/.

Visualization:

1: Line Plot

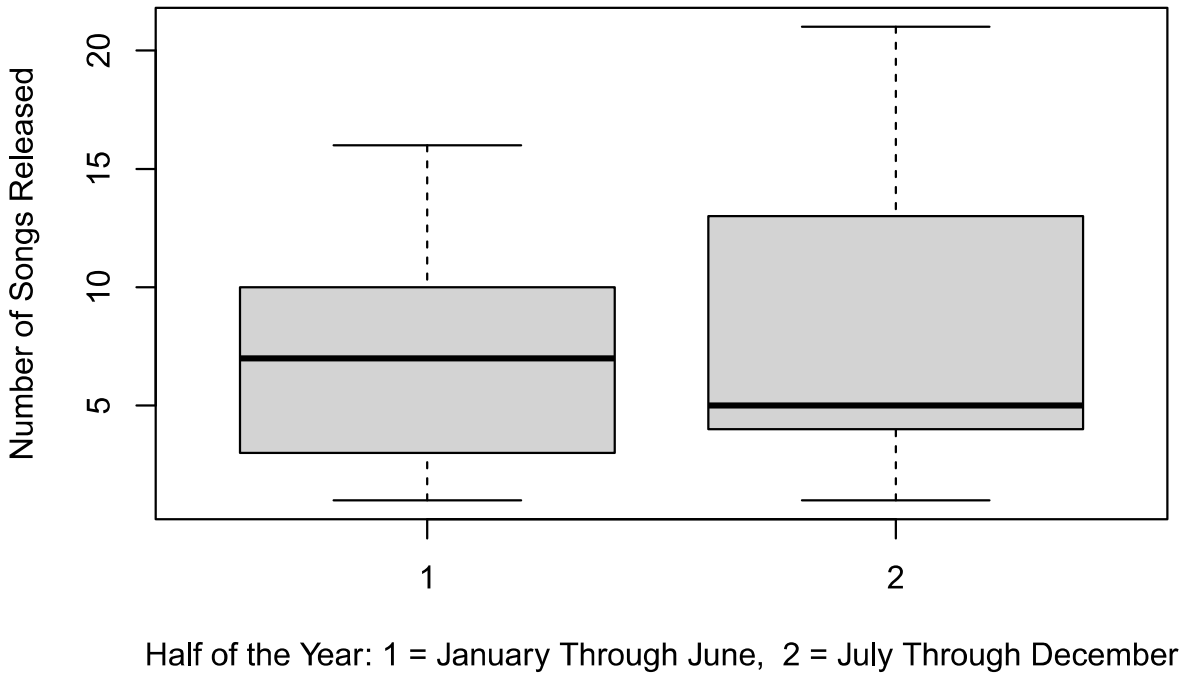
```
library(readr)
BTSData = read.csv("~/STAT 391/Spotify_BTS_Audio_OFFICIA_DATASET.csv")
BTS.music.ts = ts(BTSData[,2], start = c(2013, 1), end = c(2021, 2), frequency = 2)
plot(BTS.music.ts, xlab = "Time from June 2013 - December 2021", ylab = "Number of Songs Released")
```



In this line plot, we see that there is the data has sharp peaks and valleys. The highest peak is in the first half of 2018 (2018-06). The lowest peaks are the first half of 2014 (2014-06) and the second half of 2017 (2017-12). The data between 2015-06 through the second half of 2016 (2016-12) has a similar pattern to the data between 2019-06 to 2021-06 with the height of the peaks.

2: Box Plot

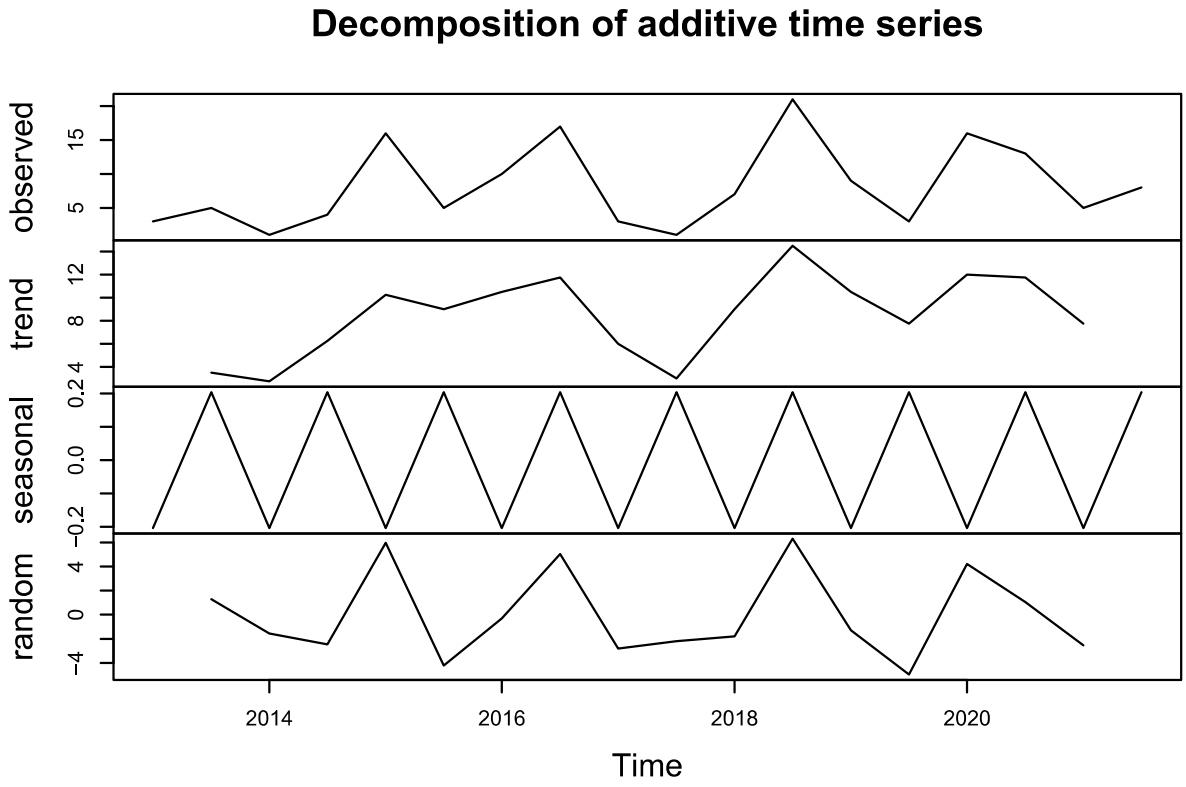
```
boxplot(BTS.music.ts~cycle(BTS.music.ts),xlab="Half of the Year: 1 = January Through June,  2 = July Th
```



In this boxplot, there are more songs released in July through December compared to January through June. The IQR of the songs released in July through December is much larger than than the IQR of January through June. January through June has a higher median value compared to July through December. Both time periods have the same amount of minimum songs (1) but the maximum amount of songs released in January through June is 16, while for July through December it is 21.

Decomposition Plot:

```
decomp.btsts = decompose(BTS.music.ts)
plot(decomp.btsts)
```



Trend: The trend of the data doesn't seem clear here. There are very large peaks and valleys in the lines that make it difficult to determine the trend of the data. The amount of songs released changes significantly, which makes it harder to tell if there's trend here.

Seasonality: The plot depicts seasonality as a sharp wave pattern, in that BTS will release a bunch of songs and then only reduce a few. They will repeat this pattern until December 2021.

Model

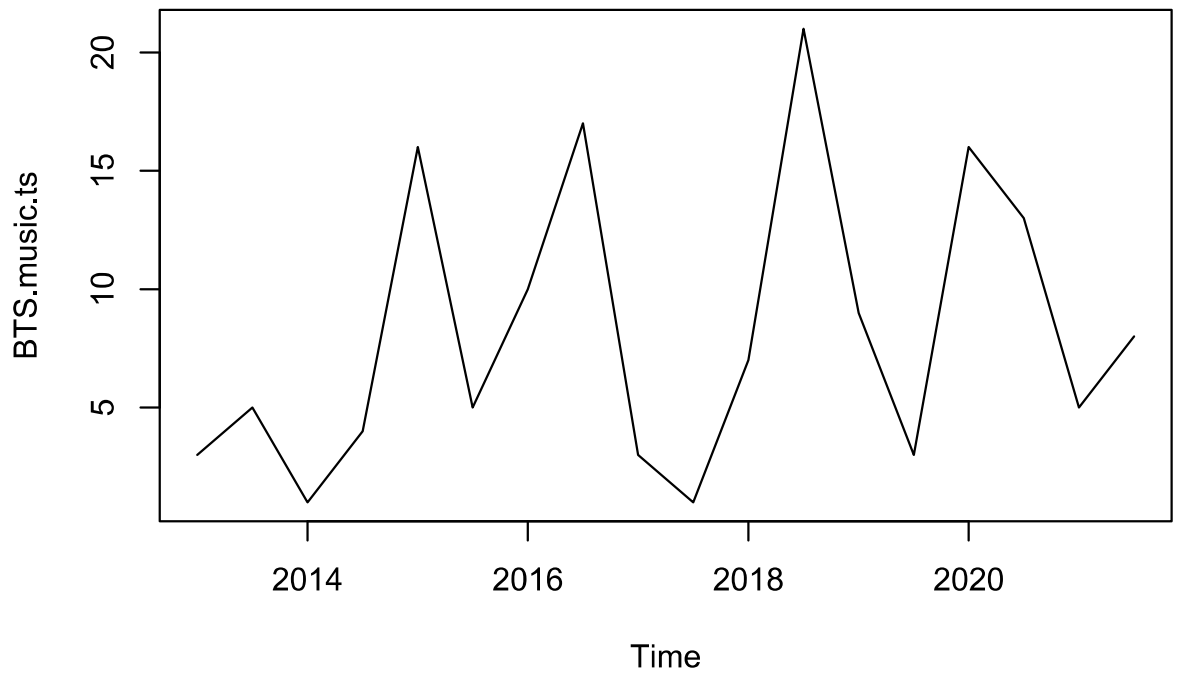
```
library(readr)
BTSData = read.csv("~/STAT 391/Spotify_BTS_Audio_OFFICIA_DATASET.csv")
library(tseries)
```

Load data

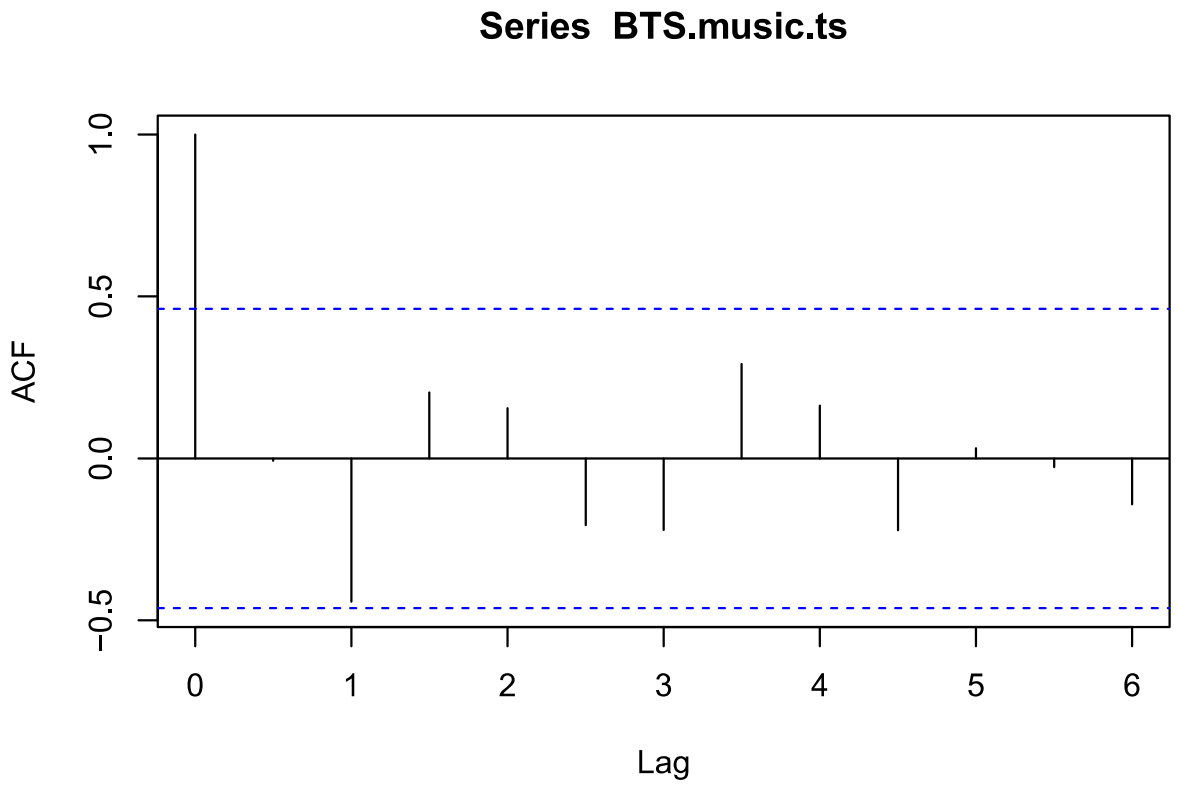
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

Check Stationarity

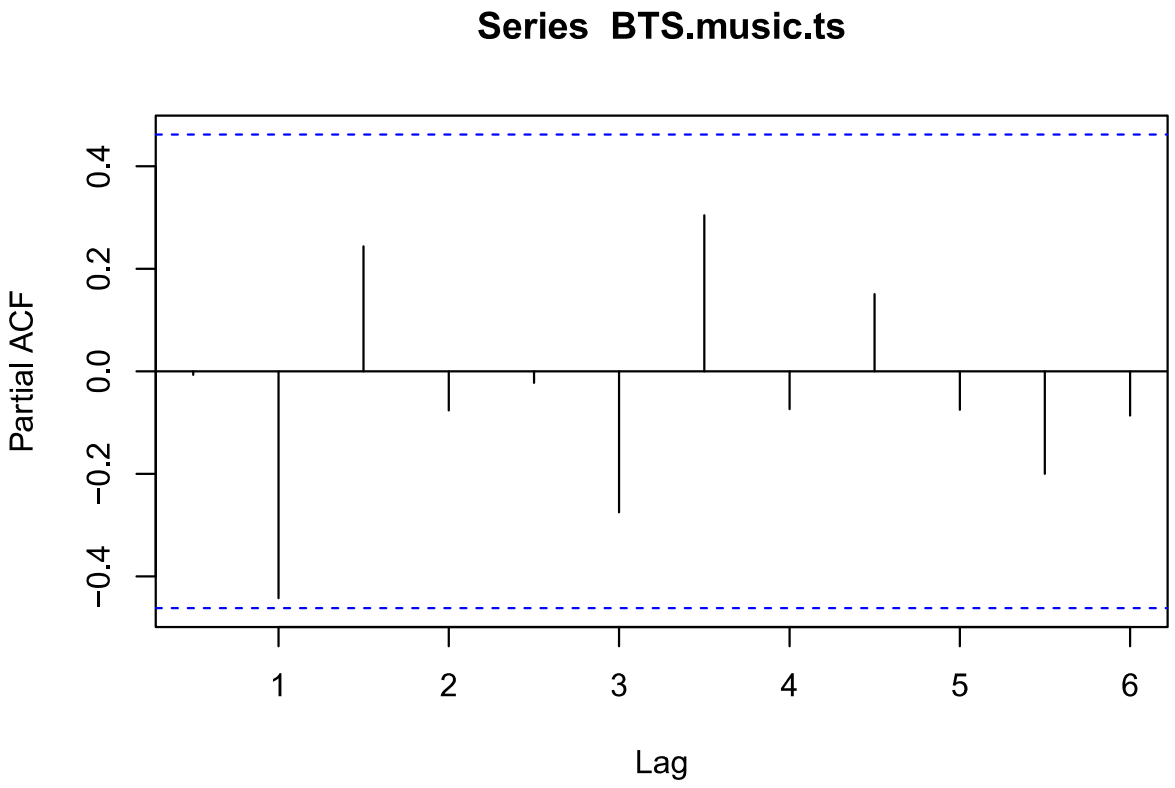
```
BTS.music.ts = ts(BTSData[,2], start = c(2013, 1), end = c(2021, 2), frequency = 2)
plot(BTS.music.ts)
```




```
acf(BTS.music.ts)
```



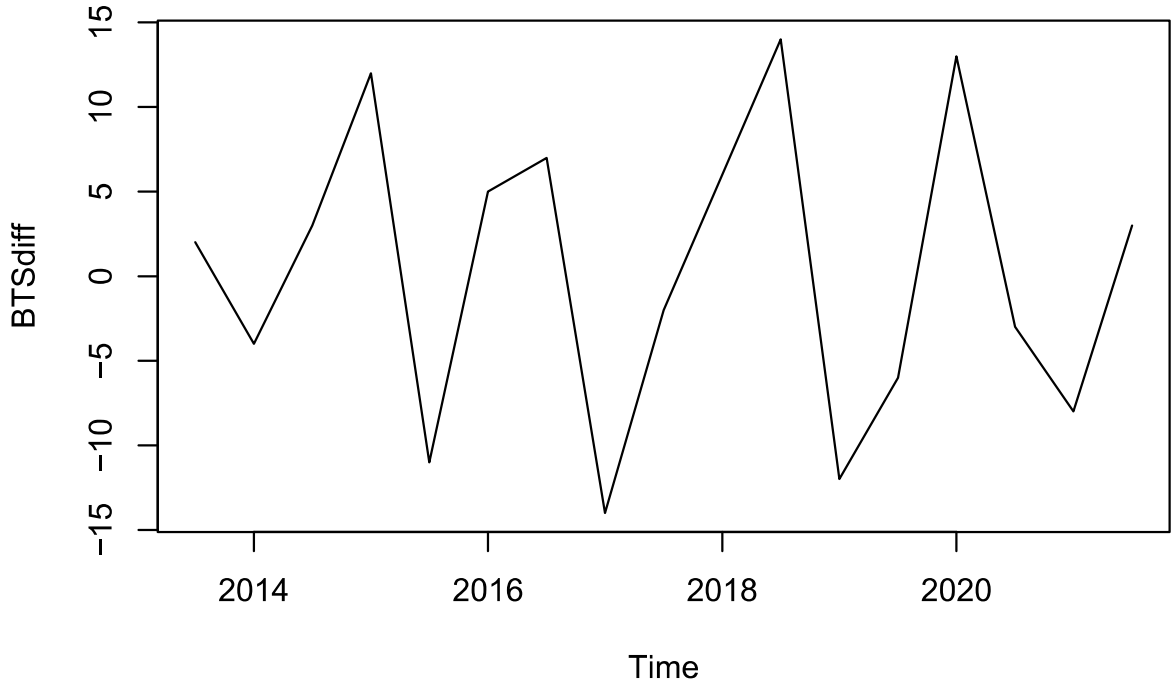

```
pacf(BTS.music.ts)
```




```
adf.test(BTS.music.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  BTS.music.ts
## Dickey-Fuller = -2.677, Lag order = 2, p-value = 0.3145
## alternative hypothesis: stationary
```

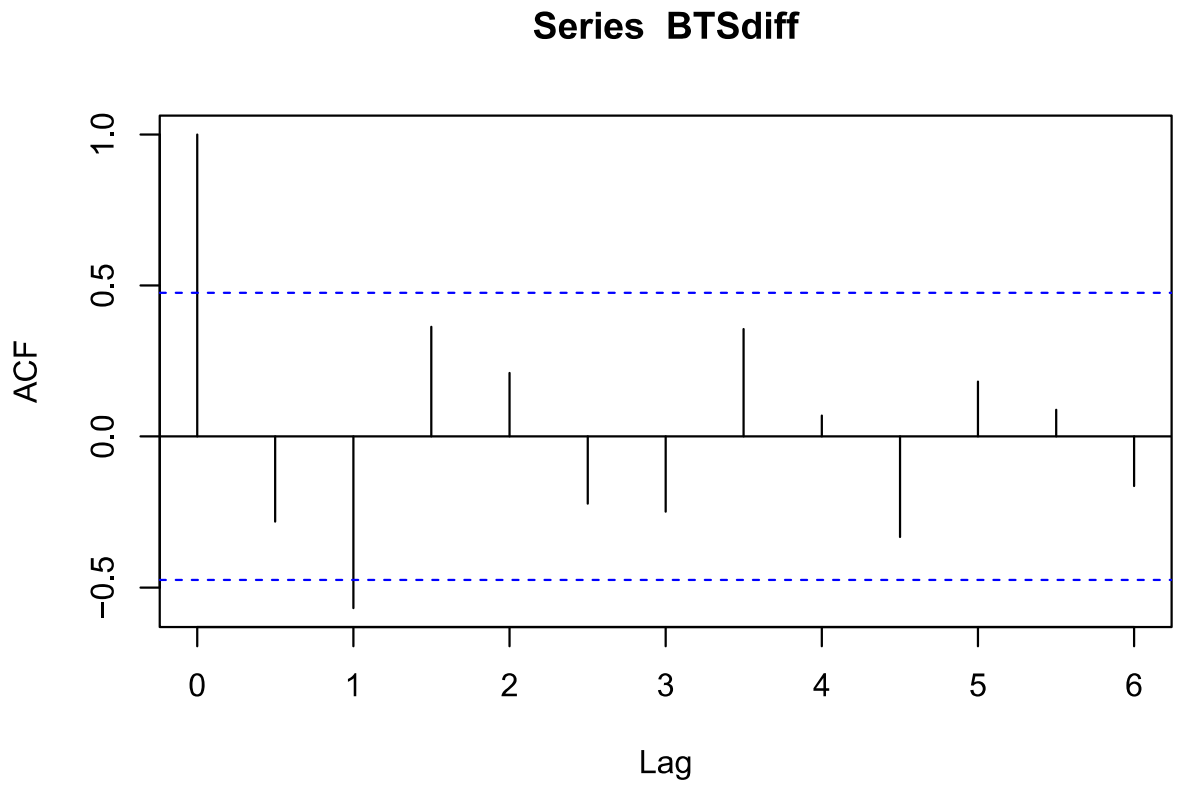
```
BTSdiff = diff(BTS.music.ts)
plot(BTSdiff)
```



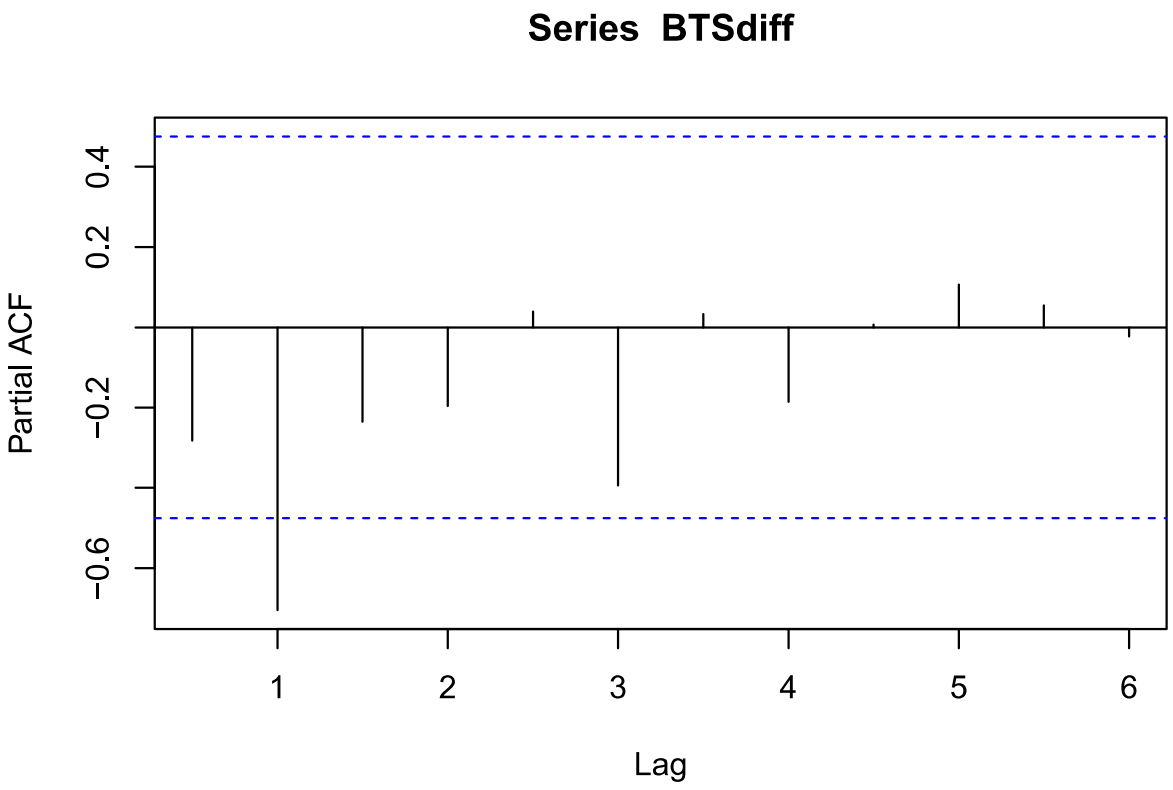

```
adf.test(BTSdiff)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  BTSdiff
## Dickey-Fuller = -3.7349, Lag order = 2, p-value = 0.04037
## alternative hypothesis: stationary
```

```
acf(BTSdiff)
```




```
pacf(BTSdiff)
```

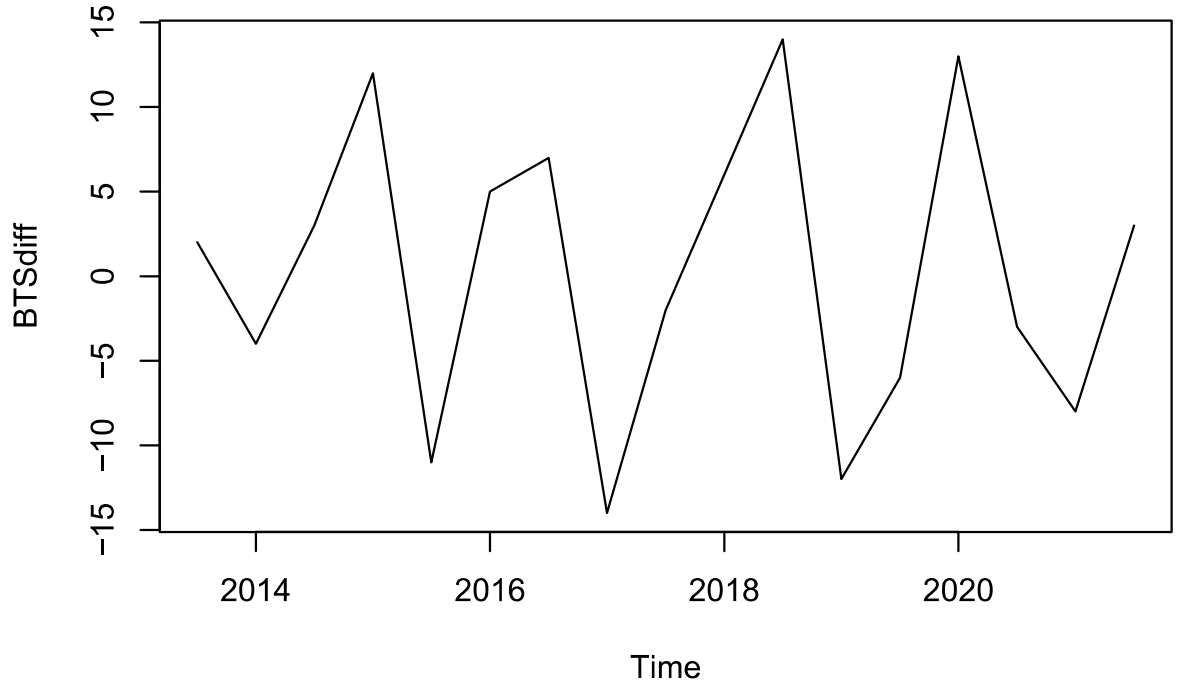


When looking at the data from the original plot, the data doesn't appear stationary because there doesn't seem to be a consistent mean value over time. Using the augmented Dickey-Fuller Test, the p-value given is .3145, so we would fail to reject the null hypothesis that the data is non-stationary. The ACF/PACF tests also support that the data is white noise and that songs released are not dependent on time in this dataset.

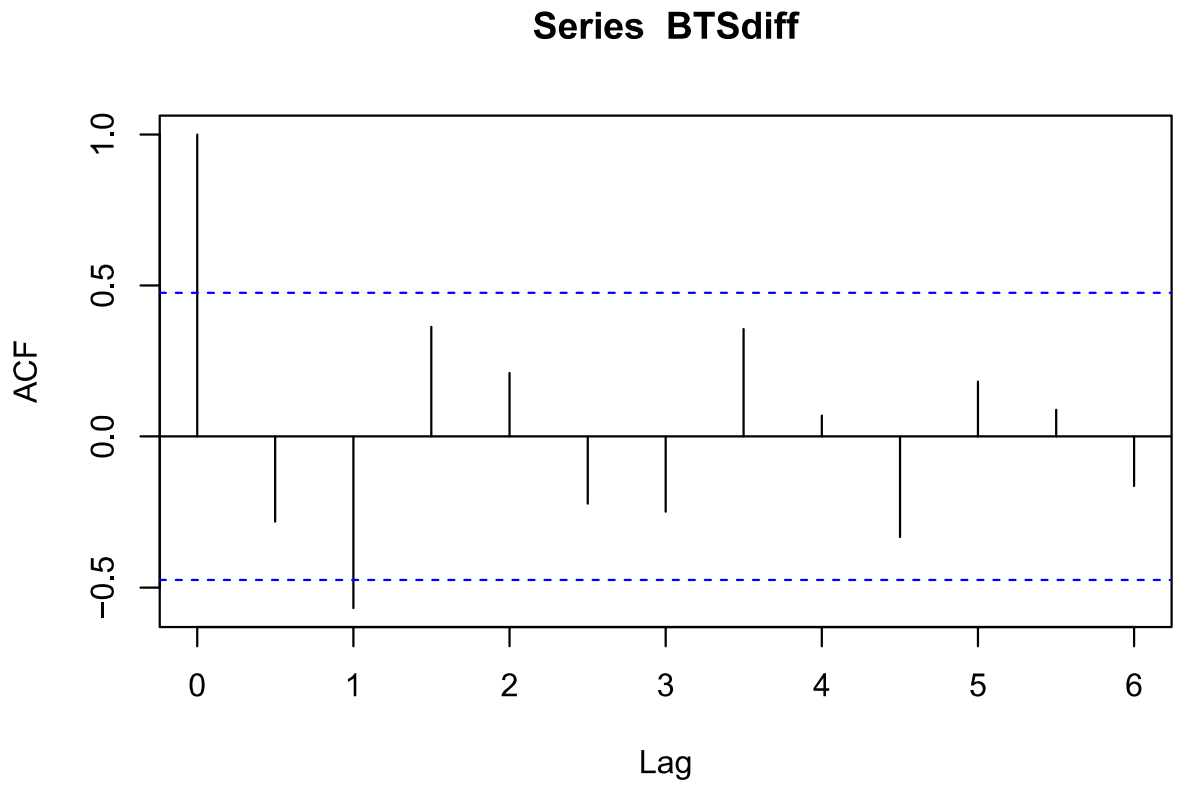
After doing a transformation, the data appears more stationary. The ACF has a sine like pattern and the 2nd lag is significant. The PACF has exponential decay but appears more unclear compared to graphs that are clearly stationary. Using the Augmented Dickey-Fuller Test, the p-value given is .04, which is smaller than .05 so we would reject the null hypothesis that the data is not stationary.

Fit Model:

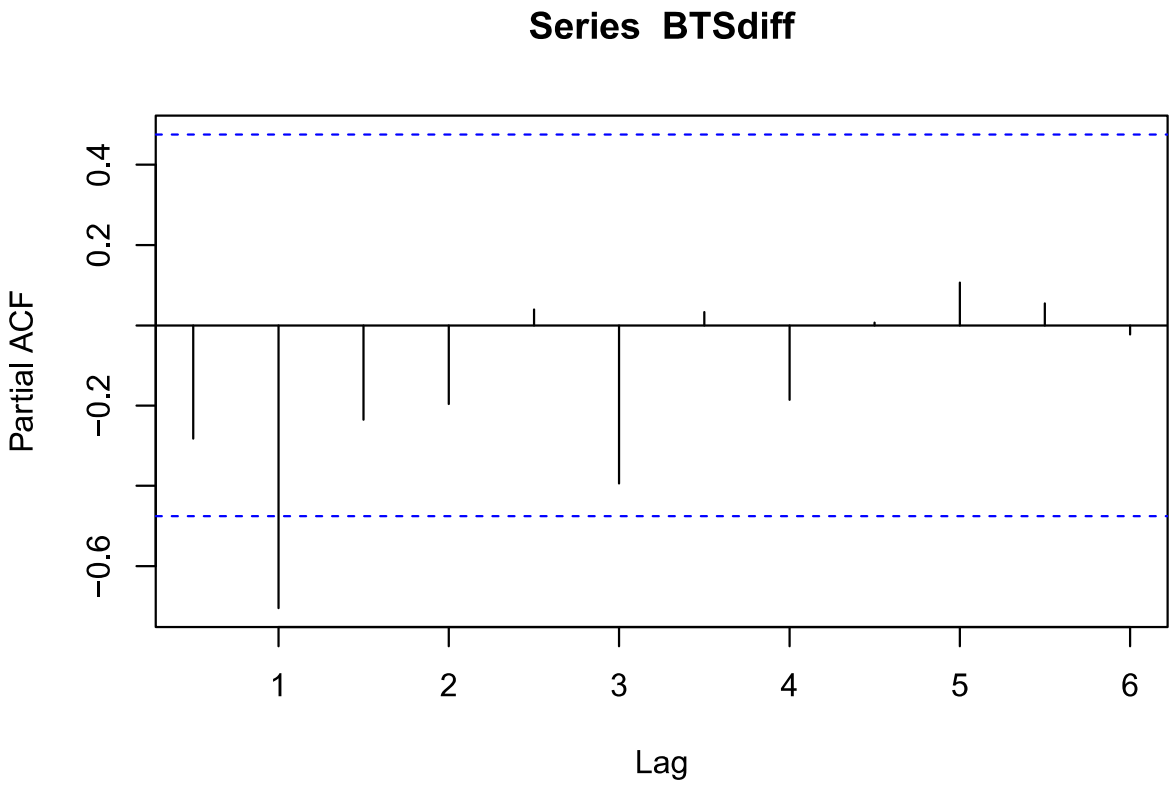
```
BTSdiff = diff(BTS.music.ts)
plot(BTSdiff)
```




```
acf(BTSdiff)
```




```
pacf(BTSdiff)
```



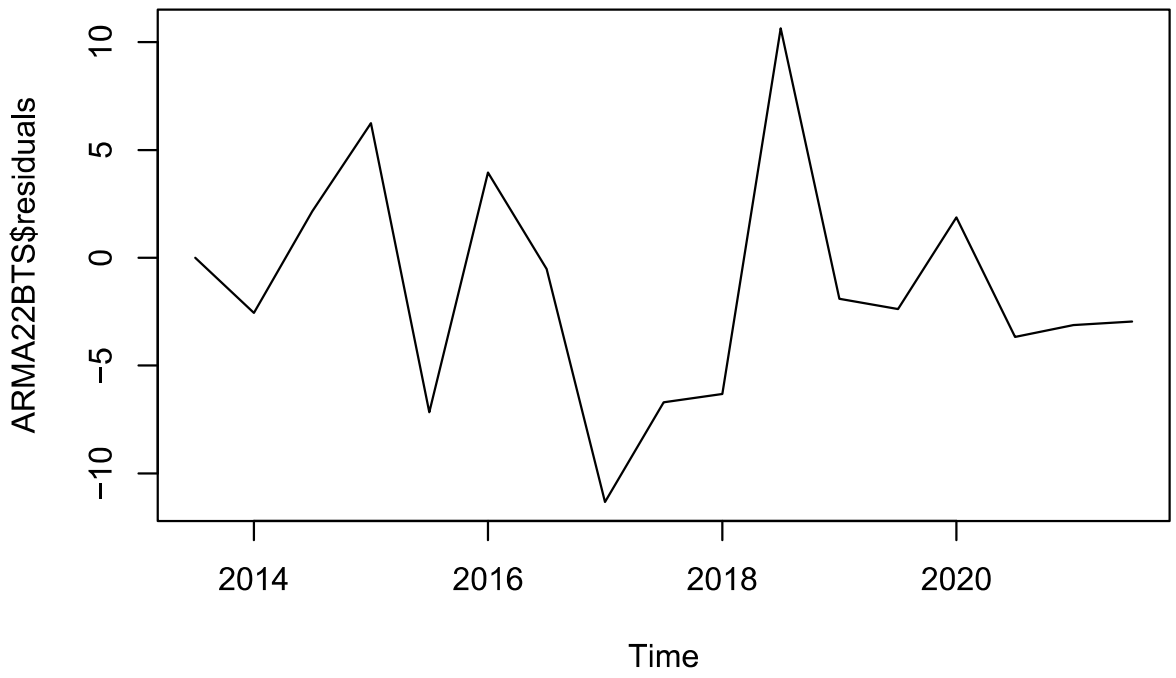

```
ARMA22BTS = arima(BTSdiff, order = c(2,1,2), include.mean = F)
```

The ACF has a sine like pattern and the 2nd lag is significant. The PACF has exponential decay but appears more unclear compared to graphs that are clearly stationary.

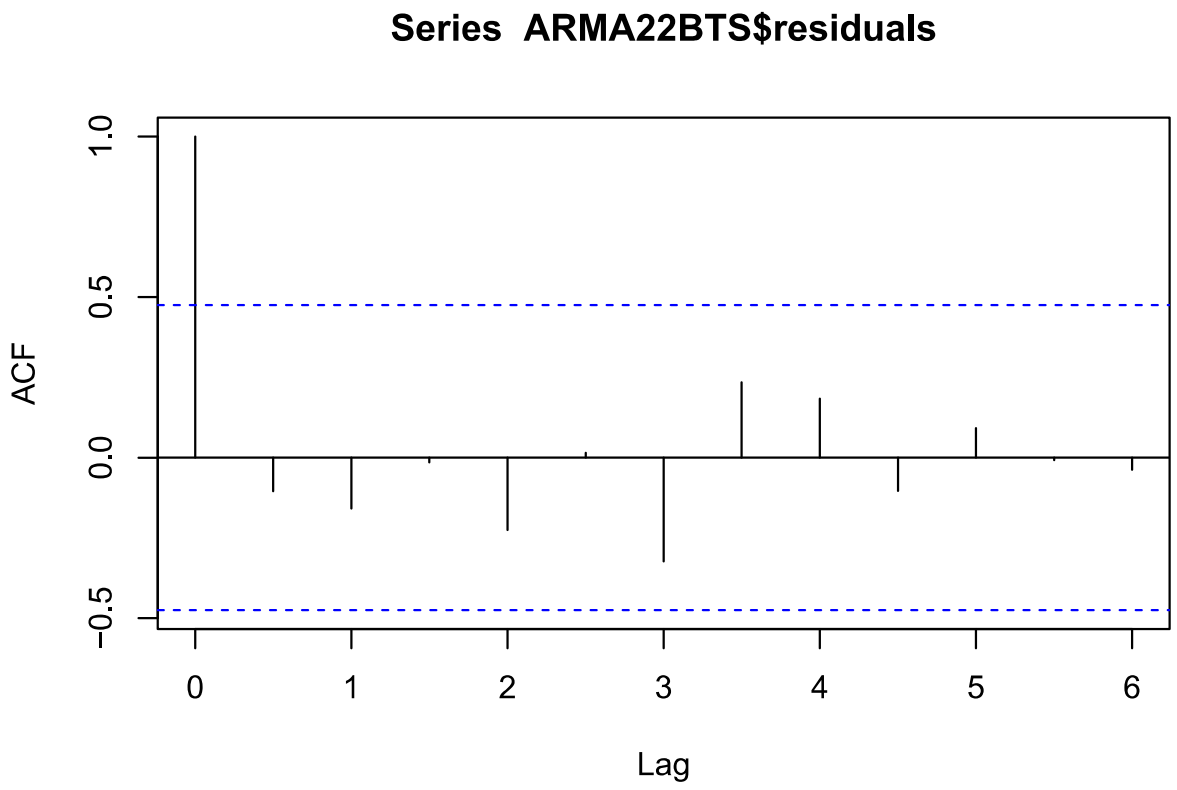
With this model, I am predicting that this is an AR(2) or an ARMA(2,2) model. However, it's also important to add that the data was differenced once for future reference.

Assess Model Fit

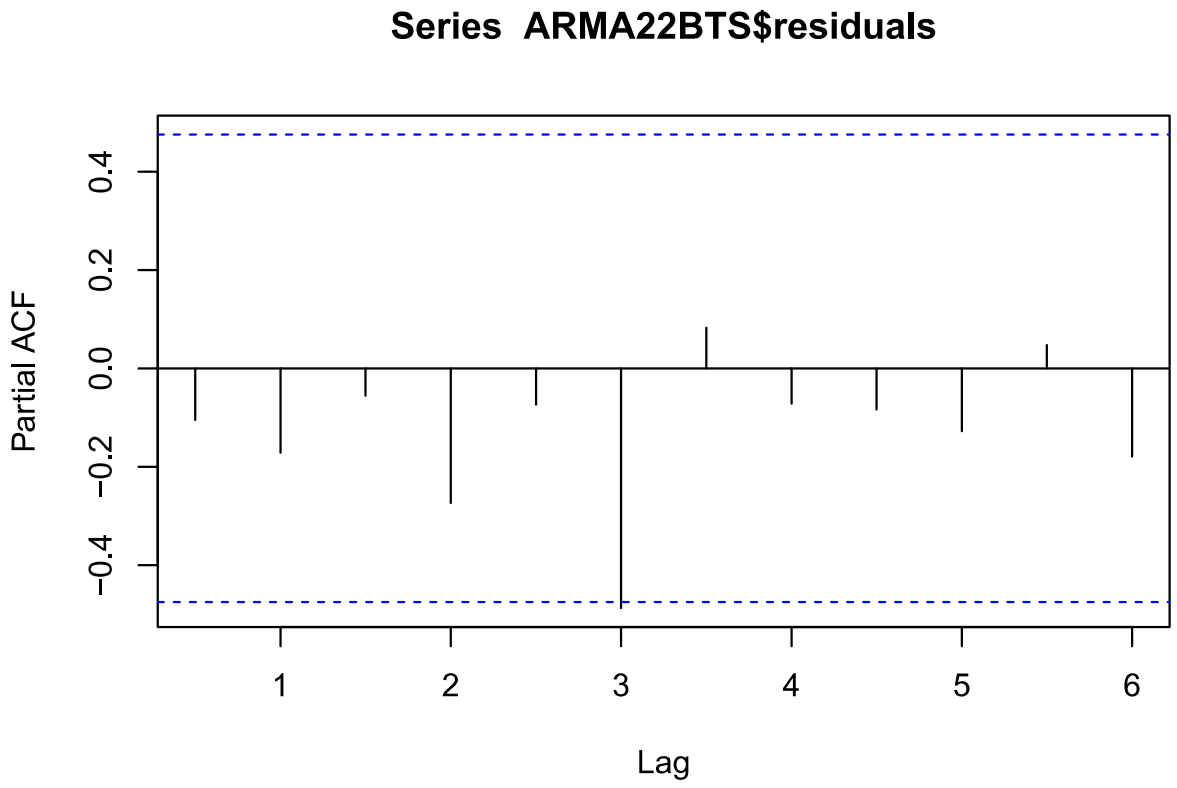
```
plot(ARMA22BTS$residuals)
```




```
acf(ARMA22BTS$residuals)
```




```
pacf(ARMA22BTS$residuals)
```




```
shapiro.test(ARMA22BTS$residuals)
```

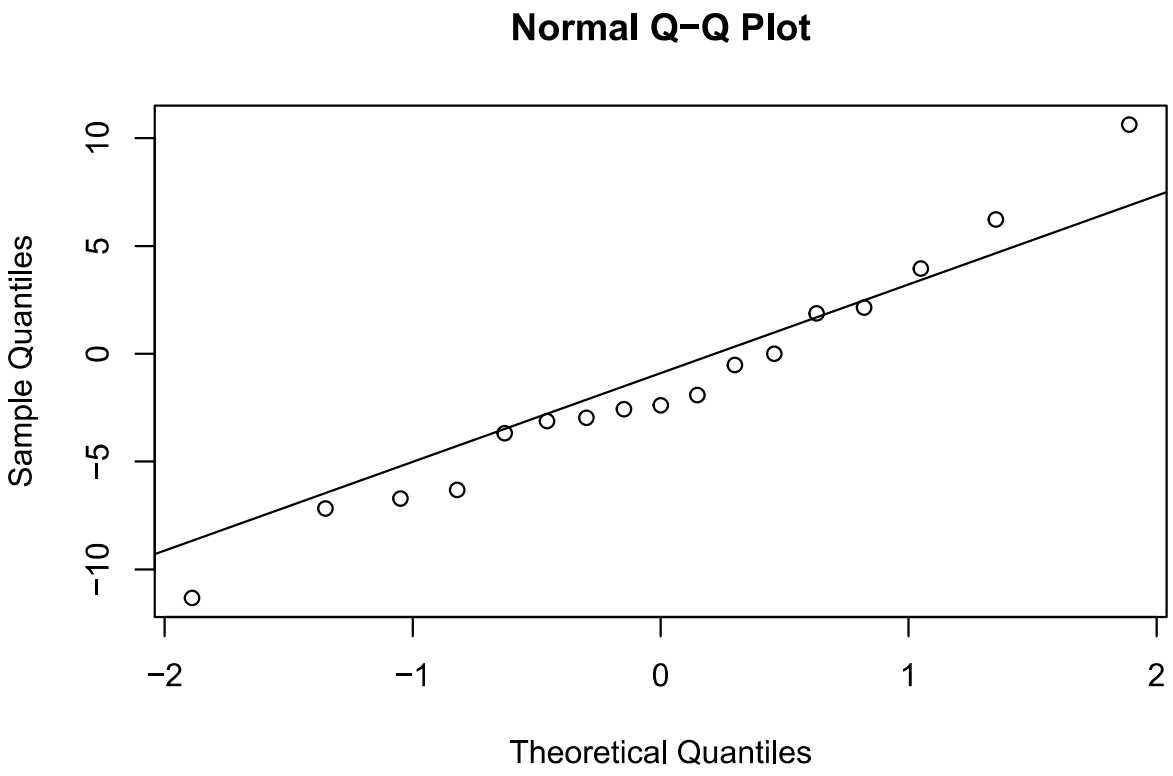
```
##
##  Shapiro-Wilk normality test
##
## data:  ARMA22BTS$residuals
## W = 0.97622, p-value = 0.9152
```

```
Box.test(ARMA22BTS$residuals, lag=16, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  ARMA22BTS$residuals
## X-squared = 9.1043, df = 16, p-value = 0.9091
```

```
hist(ARMA22BTS$residuals)
```


```
qqnorm(ARMA22BTS$residuals)
qqline(ARMA22BTS$residuals)
```



When looking at the residuals, there doesn't appear to be any trend in the data. There are prominent peaks in 2017 and the second half of the year in 2018. However, those two spikes are of interest compared to the rest of the graph.

The ACF/PACF plots have white noise as there is nothing significant about them.

The Shapiro-Wilks test gives a p-value of .9152, which means that we fail to reject the null hypothesis that the data tested is normally distributed.

The Ljung-Box test gives a p-value of .9091. Because there are less than 20 data points in my dataset, I chose a lag value of 16. I have 18 points and chose a lag value that was slightly less than 18. Because the p-value is .9091, we fail to reject the null hypothesis that the data tested is independently distributed.

The histogram looks normally distributed with slight right-skewedness. However, it is normally distributed for the most part.

The qqnorm and qqline do not line up exactly. The data will always have variations on a qqplot (probability plot). The data points in between the tails are close to the qqline. The line is relatively straight. It does appear normal.

Although it looks like an ARMA (2,2) model is a good fit with the data, it is helpful to check other models to see which is better.

Assess Model Fit: Comparing Models

```
ARMA22BTS = arima(BTSdiff, order = c(2,1,2), include.mean = F)
AR2BTS = arima(BTSdiff, order = c(2,1,0), include.mean = F)

AIC.to.AICC <- function (aic, n, pq1) {
 aic - 2 * pq1 + 2 * n * pq1 / (n - 1 - pq1)
}

ARMA22BTS$aic
```

```
## [1] 116.5813
```

```
AR2BTS$aic
```

```
## [1] 121.5856
```

```
AIC.to.AICC(ARMA22BTS$aic, 18, 5)
```

```
## [1] 121.5813
```

```
AIC.to.AICC(AR2BTS$aic, 18, 3)
```

```
## [1] 123.2999
```

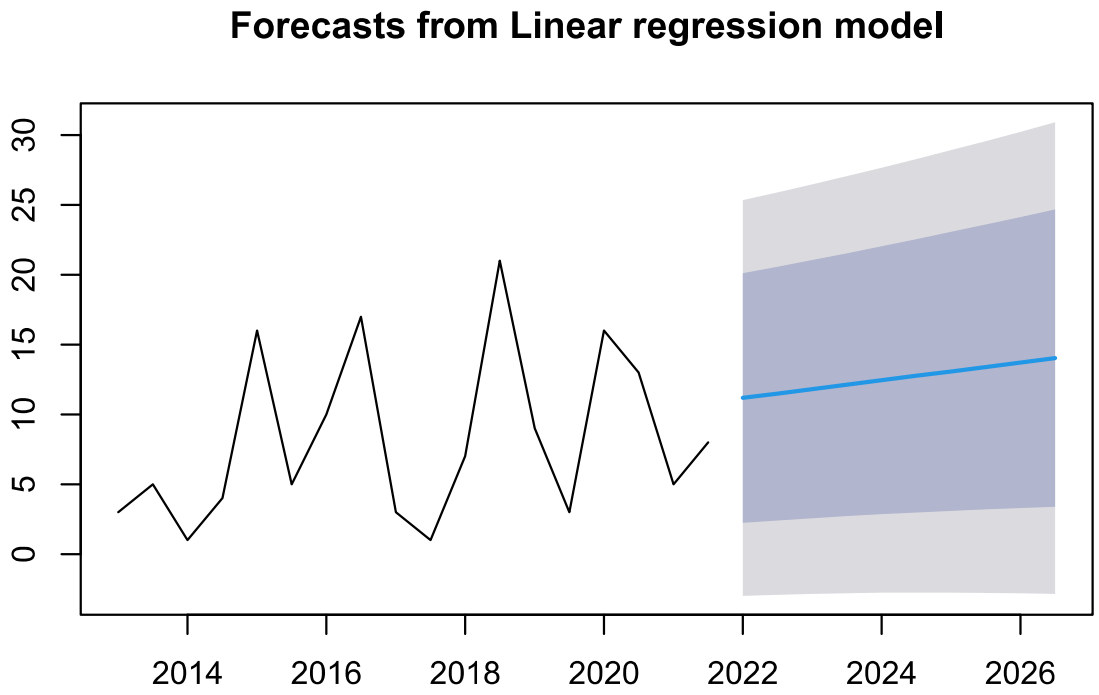
Here in fact, the ARMA (2,2) model is a good fit for the data. The AIC and the AICc scores are smaller with the ARMA (2,2) data compared to the AR(2) model.

Forecast

```
library(forecast)
BTSFit = tslm(BTS.music.ts ~ trend)
BTSFit
```

```
##
## Call:
## tslm(formula = BTS.music.ts ~ trend)
##
## Coefficients:
## (Intercept)        trend
##      5.1569       0.3168
```

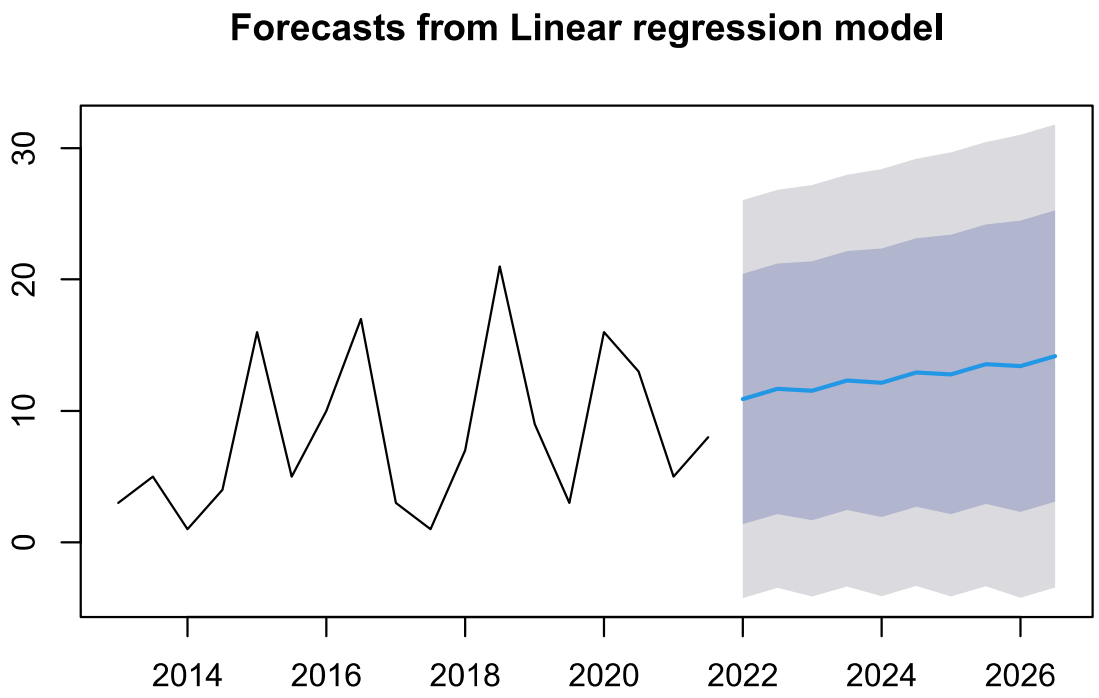
```
forecastBTSFit = forecast(BTSFit, h = 10)
plot(forecastBTSFit)
```




```
BTSFitting = tslm(BTS.music.ts ~ trend + season)
BTSFitting
```

```
##
## Call:
## tslm(formula = BTS.music.ts ~ trend + season)
##
## Coefficients:
## (Intercept)        trend      season2
##      4.9653       0.3125       0.4653
```

```
forecastBTSFitting = forecast(BTSFitting, h = 10)
plot(forecastBTSFitting)
```




```
forecastBTSFitting
```

```
##         Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## 2022.00       10.90278 1.369370 20.43619 -4.254534 26.06009
## 2022.50       11.68056 2.147148 21.21396 -3.476756 26.83787
## 2023.00       11.52778 1.668770 21.38679 -4.147211 27.20277
## 2023.50       12.30556 2.446548 22.16456 -3.369433 27.98054
## 2024.00       12.15278 1.922321 22.38323 -4.112783 28.41834
## 2024.50       12.93056 2.700099 23.16101 -3.335005 29.19612
## 2025.00       12.77778 2.134823 23.42073 -4.143620 29.69918
## 2025.50       13.55556 2.912601 24.19851 -3.365842 30.47695
## 2026.00       13.40278 2.310855 24.49470 -4.232442 31.03800
## 2026.50       14.18056 3.088633 25.27248 -3.454664 31.81577
```

Looking only at trend, I predict that BTS will release more music and that they will release more songs over time.

Taking into account seasonality, the data follows the pattern similar to previous years. Through forecasting, I predict that BTS will continue to release more songs, but the amount of songs they release will have a wave pattern.

The exact points are below:

January-June 2022: 10.90 songs

July-December 2022: 11.68 songs

January-June 2023: 11.53 songs

July-December 2023: 12.31 songs

January-June 2024: 12.15 songs

July-December 2024: 12.93 songs

January-June 2025: 12.78 songs

July-December 2025: 13.56 songs

January-June 2026: 13.4 songs

July-December 2026: 14.18 songs

I used 10 as the amount of values I wanted to forecast because each member of BTS has a contract with HYBE until 2026. I had the forecast go through 2026 to observe the trend/seasonality. Between 2023-2026, BTS as a group will not release any new music however.

Concluding Statements and Recommendations

Looking at only new released music from BTS's debut until the end of 2021, it was challenging to fit a time series model with only 18 points of data. I differenced the data so that it was more stationary and easier to work with. Using ACF/PACF plots and the augmented Dickey-Fuller test, I concluded that the data was stationary and that the data fit an ARMA(2,2) model. Using AIC and AICc with other models such as an AR(2) for the data, I found that the best model fit was the ARMA(2,2) model. Using the forecasting package, I forecasted the data for the next 10 points (2022-2026). With limited data like I had, it made it hard to truly determine if BTS's song release frequency was dependent on time. However, differencing the data did show that there is a relationship between BTS's song quantity relase and time. In the future, I would like to continue observing the amount of songs that BTS has released, but I would also like to look into each member's solo projects too. Since announcing their military enlistment plans in October 2022, Jin, SUGA, Jimin, and RM have all released new albums and in the future, I'd like to see how that influences the amount of songs released over time. In this dataset, re-released songs were taken out and I wonder how that would've influenced the data.