Spotify Songs Analysis & Forecast

Taylan Kabbani

January 27, 2021

Problem Definition

- **Project Objective:** Analyzing the trends of Turkish songs on *Spotify* and forecasting the popularity of songs for the following week.
- **Determining what to forecast:**
  1. Point forecasts for the number of weekly streams of top 100 songs in week 01/07/2021.
  2. predict whether specific 200 songs (in week 01/07/2021) will be in the top 200 list in the list of of 200 top songs of week 28/01/2021.
- **Data:** *Spotify* publishes weekly number of streams for each song in the top 200 list for each county at spotifycharts.com. Here, country will be set to Turkey as we are only interested in Turkish songs.
- **Task Difficulty:** Due to low forecast accuracy, a large number of songs, shortening life cycle of songs, and sudden and frequent changes in music tastes of the society, the task is considered a difficult one.
- **Programming Language:** R programming language is being used in this project. Project's files and code scripts can be found on this Github repository

Gathering information

Spotify publishes weekly and daily streams for each song in the top 200 to be downloaded on their website. However, manually downloading this data for each week is tedious and time-consuming since we want to retrieve all weeks since 2016 up until now. Using *rvest* package in R, we can scrape the website and retrieve the data we want. DataScraping.R file does the following:

1. Create a URL sequence function that will feed our scraper the right directions to find WHERE the data is.
2. Create a scraper function that will find WHAT attributes we want into a scrape.
3. Data Wrangling to reshape the data into a tibble using *dplyr and tibble* packages.
4. Download the data as .xlsx file. (see file here)

Since our time-series (songs) does not share the same time horizon, I divide the top 200 songs of week 07/01/2021 into three groups based on the number of data points available to retrieve in the past (see Data Wrangling.R:

1. Very short time-series: <= 10 weeks
2. Short time-series: > 10 weeks but <= 54 weeks (less than a year)
3. Long time-series: > 54 weeks (more than one year)

Also the name of the song and the Artist of the top 100 songs for the stream forecasts are being saved.

Rank Forecast

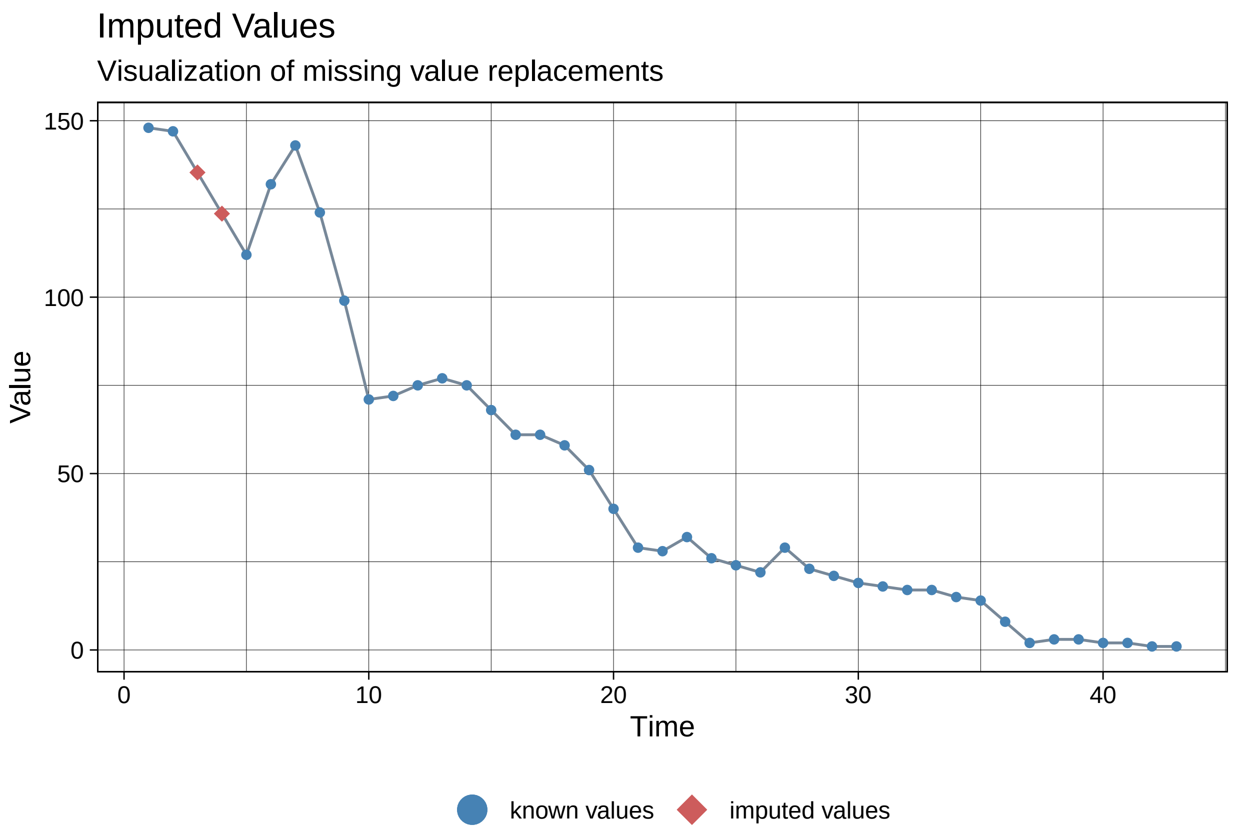
Preliminary (exploratory) analysis

Because we have many time-series, I examined few songs from each group of the three groups I created (explained in the previous section) to get a better understanding of the data patterns, check if there is a significant trend? is seasonality important? is there evidence of the presence of business cycles?. The following is an example of the analysis performed on one song from short time-series group.

```r
library(readxl)
library(tidyverse)
library(ggfortify)
library(ggplot2)
library(tsbox)
library(imputeTS)
library(cowplot)
library(forecast)

SpotifyData <- read_excel("Data/SpotifyData_full.xlsx")

# Extracting specific song streams and return it as ts object
FindSong <- function(SongName, ArtistName){
  song_Rank <- as.data.frame(SpotifyData
                             %>% filter(song == SongName & Artist == ArtistName)
                             %>% select(Date,Rank))

  # ts_ts will convert df to time series
  song_Rank <- ts_ts(ts_c(song_Rank))
  song_Rank
}
song2 <- FindSong('Seni Dert Etmeler', 'Madrigal')
song2
```

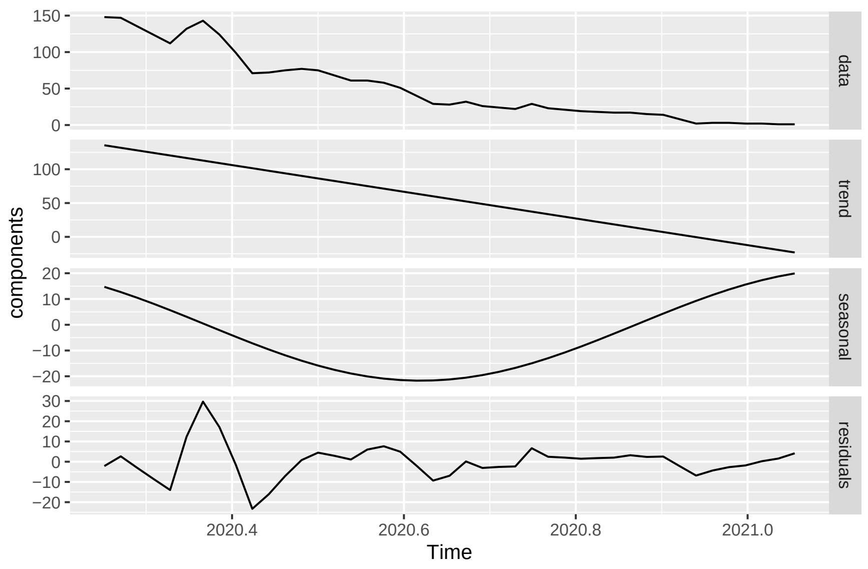
```
## Time Series:
## Start = 2020.25136612022
## End = 2021.05479452055
## Frequency = 52.2759713034483
##  [1] 148 147  NA  NA 112 132 143 124  99  71  72  75  77  75  68  61  61  58  51
## [20]  40  29  28  32  26  24  22  29  23  21  19  18  17  17  15  14   8   2   3
## [39]   3   2   2   1   1
```

For some songs there are missing values for some weeks in the time series, as in the song above, to handle missing values *Imputation by Interpolation* method is used to fill missing values. This method is appropriate for time series with strong trend as we are expecting a trend in our time-series.

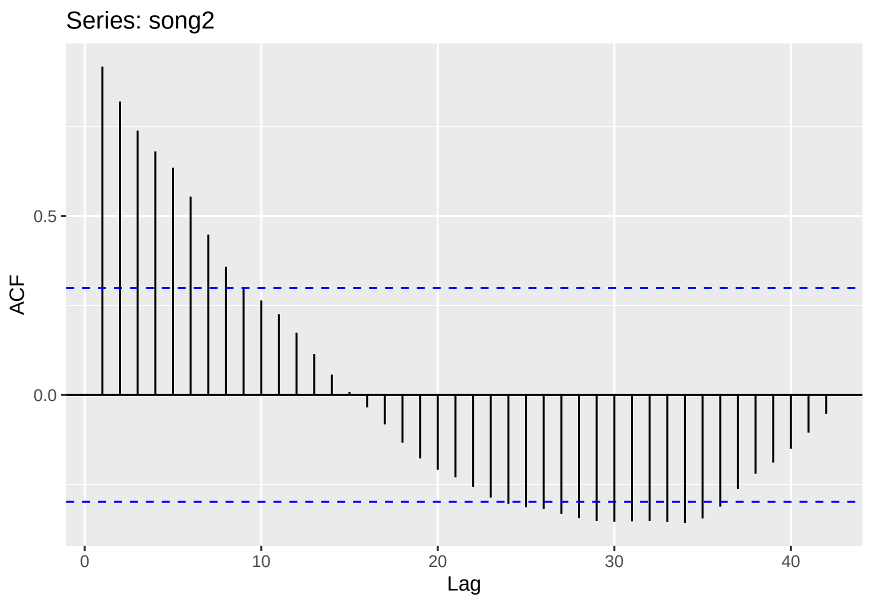


In our time-series data, the song streams/rank has been recorded at regular intervals (weekly), however for many of these songs (short and very short groups) have less than one year (42 weeks for the example here), therefore normal decomposition methods will not be useful since time series should have data points not less than two periods. It is possible to use a linear regression model to decompose a time series into a trend and seasonal components, and then some smoothness assumptions on the seasonal component allow a decomposition with fewer than two full years of data.

```r
# Approximating the seasonal pattern using Fourier terms with a few parameters.
decompose_song2 <- tslm(song2 ~ trend + fourier(song2, 1))
# Trend component as moving average function
trend <- coef(decompose_song2)['(Intercept)'] + coef(decompose_song2)['trend']*seq_along(song2)
e <- residuals(decompose_song2)
# Additive model is used to decompose the time series data
seasonality <- song2 - trend - e
components <- cbind(
  data = song2,
  trend = trend,
  seasonal = seasonality,
  residuals =e
  )
autoplot(components,facet=TRUE)
```



```r
ggAcf(song2)
```



```r
# Finding the best lambda
(lambda <- BoxCox.lambda(song2))
```

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

From the ACF plot and the decomposition of the time series, we conclude that:

1. The time series is non stationary because of the significant spikes in the ACF plot.
2. Strong trend and cyclical components, as the trend component and the exponential decreasing in the ACF support the assumption.
3. There might be seasonality in the data, however, the short time-series we are dealing with showed no clear seasonal pattern.
4. There is no transformation required for the data (as seasonality is not a big concern here)

Choosing and evaluating models

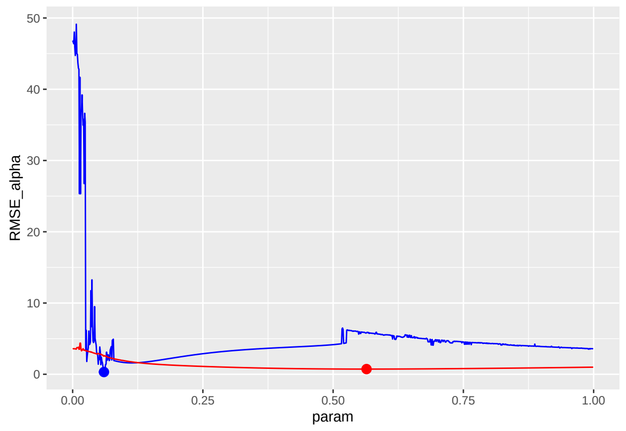
In this section I choose the best models to forecast the next rank of a song based on the analysis conducted in the previous section. Below all the methods used are explained with an example of a song that best fit it. **Data partitioning:** leaving the last observation as test data, as we are only interested in forecasting only the next data point (short-term forecasting)

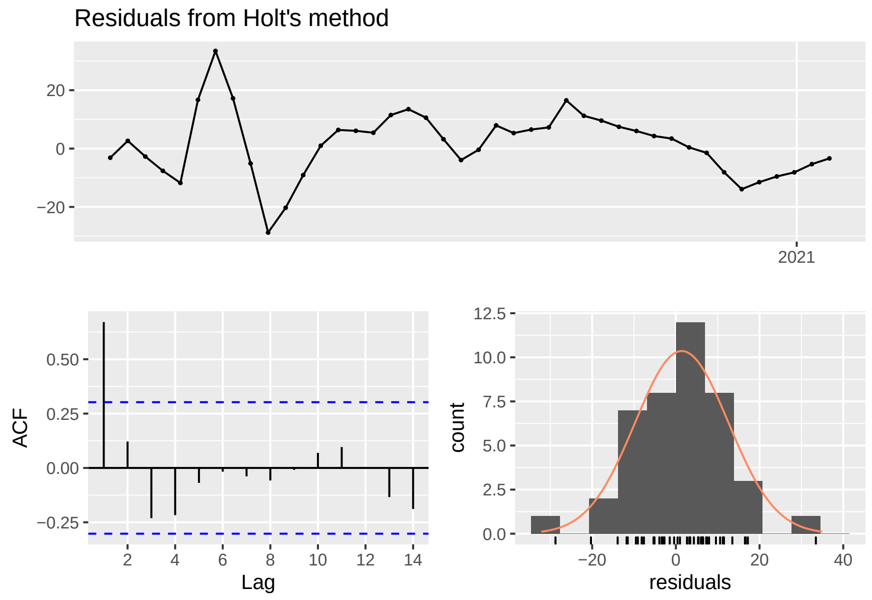
```
train <- subset(song2, end = length(song2)-1)
test <- subset(song2, start= length(song2))
```

Holt's linear trend method:

Starting with Holt's linear trend method, as we anticipate a clear trend in the data this method is a good point to start with, setting to start the initial solution by taking the average values of the first few observations. Holt's Method makes predictions for data with a trend using two smoothing parameters, alpha, and beta which correspond to the level and trend components, respectively. Holt's method parameters (alpha and beta) are being optimized using grid search.

```
Optimal_param <- function(data, show=FALSE){
  train <- subset(data, end = length(data)-1)
  validate <- subset(data, start= length(data))
    # loop through a series of betas and alphas
  param <- seq(.0001, 1, by = .001)
  RMSE_beta <- NA
  RMSE_alpha <- NA
  for(i in seq_along(param)) {
    fit_alpha <- holt(train, alpha = param[i], h = 1)
    RMSE_alpha[i] <- accuracy(fit_alpha, validate)[2,2]
    fit_beta <- holt(train, beta = param[i], h = 1)
    RMSE_beta[i] <- accuracy(fit_beta, validate)[2,2]
  }
  # convert to a data frame and idenitify min alpha and beta value
  df <- data_frame(param, RMSE_alpha, RMSE_beta)
  alpha.opt <- filter(df, RMSE_alpha == min(RMSE_alpha)) %>% select(param,RMSE_alpha)
  beta.opt <- filter(df, RMSE_beta == min(RMSE_beta)) %>% select(param,RMSE_beta)
  # plot RMSE vs. alpha
  if (show==TRUE) {
      plot <- ggplot(df, aes(param,RMSE_alpha)) +
        geom_line(color='blue')+
        geom_line(aes(param,RMSE_beta),color = 'red')+
        geom_point(data = alpha.opt, aes(param, RMSE_alpha), size = 3, color = "blue")+
        geom_point(data = beta.opt, aes(param, RMSE_beta), size = 3, color = "Red")
      return(plot)} else{return(list(alpha.opt,beta.opt))}
}
Optimal_param(song2,show=TRUE)
```





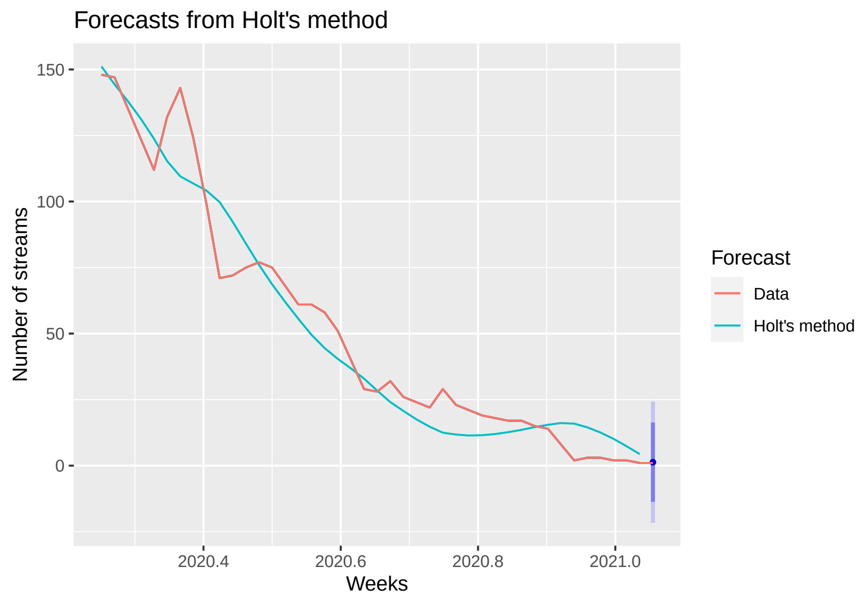
```
##
##  Ljung-Box test
##
## data:  Residuals from Holt's method
## Q* = 26.302, df = 4, p-value = 2.75e-05
##
## Model df: 4.   Total lags used: 8
```

The model passes residuals check:

1. The residuals are uncorrelated(There are a few significant spikes in the ACF)
2. The residuals have zero mean.
3. The variation of the residuals stays much the same across the historical data, apart from the one outlier.
4. Residuals are normal
5. The large negative residual is a result of the unexpected jump.
6. The model fails the Ljung-Box test. The model can still be used for forecasting.

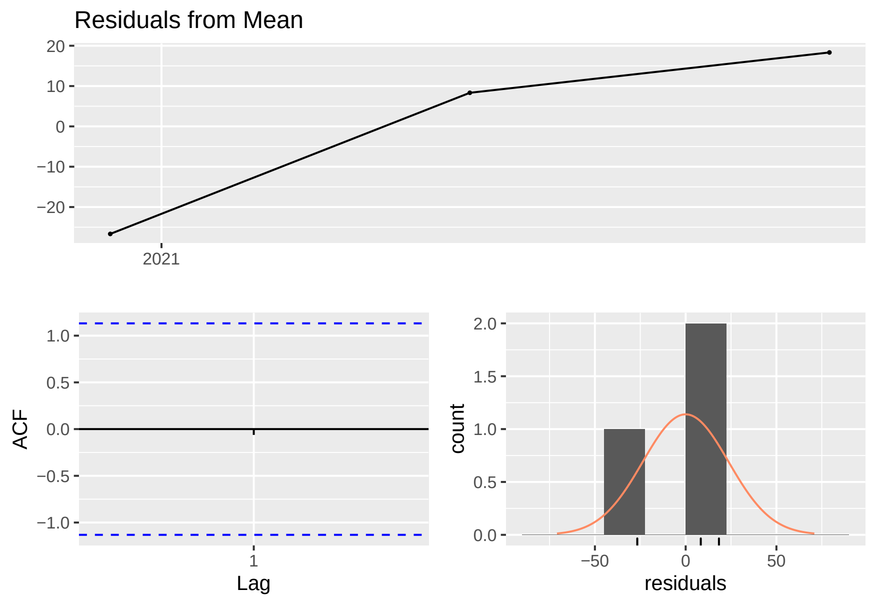
By checking the model accuracy we have about a 24.08 error according to MAPE rate and test-RMSE equals to 0.24

```
##                   ME  RMSE  MAE    MPE  MAPE MASE ACF1
## Training set  1.41 11.14 8.76 -50.36 75.49  NaN 0.67
## Test set     -0.32  0.32 0.32 -31.86 31.86  NaN   NA
```



Average method:

The average method was effective in forecasting time-series with few points such as the song showed bellow. The model passed residuals check as there is no enough data point to interpret it. According to MAPE we have about a 0.28 error rate and test-RMSE = 0.33



```
##
##  Ljung-Box test
##
## data:  Residuals from Mean
## Q* = NA, df = 3, p-value = NA
##
```

```
## Model df: 1.   Total lags used: 4
```

```
##                ME  RMSE   MAE   MPE  MAPE MASE  ACF1
## Training set 0.00 19.29 17.78 -2.91 16.01  NaN -0.06
## Test set     0.33  0.33  0.33  0.28  0.28  NaN    NA
```

Random Walk(Naive forecast):

- The naive forecasting method also performed well for some time-series with very short and short horizon.
- In addition, Random walk model with drift also has shown well performance on some songs that has very upward/down trend which takes trend into consideration by adding the difference between this period and the last period.
- The same process of residual checking and evaluation has been performed as in previous described methods, both models passed the residual checks

Pipeline for the Rank forecast

The Rankfcast_pipeline.R run the selected models on each song and choose the best one based on RMSE. Returns an Excel file as below:

```
## # A tibble: 6 x 7
##   song              Artist                Model      train_pred test  RMSE  Fcast
##   <chr>             <chr>                 <chr>      <chr>      <chr> <chr> <chr>
## 1 HIGHEST IN THE ~ Travis Scott          Holt's me~ 178        178   0     180
## 2 Affet             Müslüm Gürses         Holt's me~ 181        181   0.15  181
## 3 Birakman Dogru ~ Anil Piyanci, Zeynep~ Holt's me~ 197        197   0.27  199
## 4 LOLO              Ezhel                 Holt's me~ 187        187   0     190
## 5 Rauf              Rauf & Faik           Holt's me~ 201        199   2.16  210
## 6 Bodrum            Yüzyüzeyken Konusuruz Holt's me~ 200        200   0.01  203
```

Stream Forecast

Preliminary (exploratory) analysis

Same steps are being applied as in Rank Forecast.

Choosing and evaluating models

Unlike Rank forecasting in which we were only interested if the song in or out of the top list, in Stream foresting we need to apply more advanced forecasting models in order to capture the right stream counts.

Holt's linear trend method:

The same models (with damped trend and without) with optimizing alpha and beta parameters are applied to forecast the streams of each song. This model is particularly useful with time-series in the short group and with clear trend.

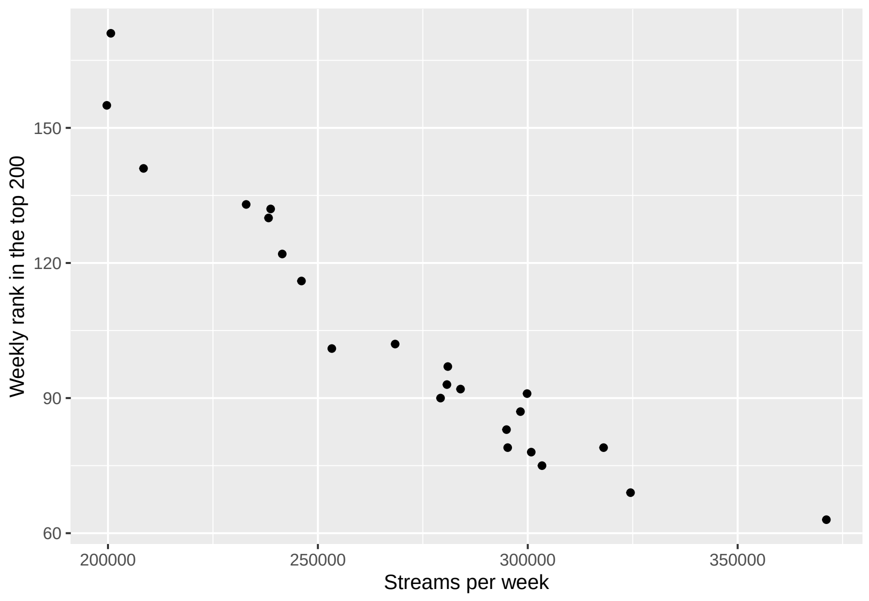
ETS(M,A,N):

By letting ETS automatically selecting the model, we get ETS(M,A,N), Holt's linear method with multiplicative errors. This model is assumed "optimal" because it minimizes RMSE, AIC, and BIC on the training data set, but does not necessarily minimize prediction errors on the test set.

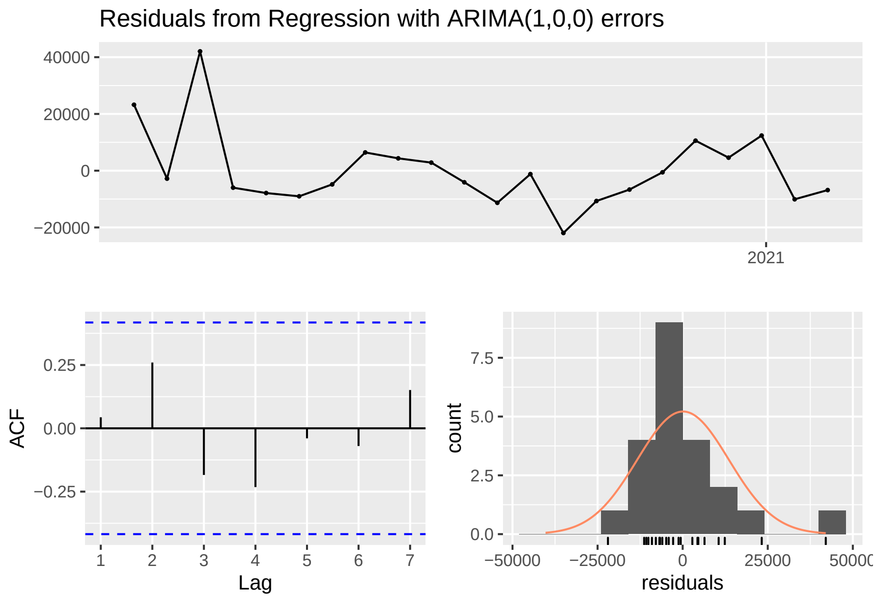
ARIMA models:

Because we have a large number of time-series to work with, auto.arime() will be used to select the model rather than manually investigating the ACF and PACF to choose Arima's coefficients, to force auto.arime() to investigate larger set of models by using the arguments stepwise=FALSE and approximation=FALSE. Two types of Arima models are being used:

- **Non-seasonal ARIMA model** using the argument seasonal=FALSE to prevent it searching for seasonal ARIMA models.
- **Dynamic regression** Using Rank as I have examined a clear correlation between Streams and Rank of the song, as the one shown below. This prediction method is heavily dependent on the quality of the Rank prediction I performed in the last section (*Ex-post forecasts*)

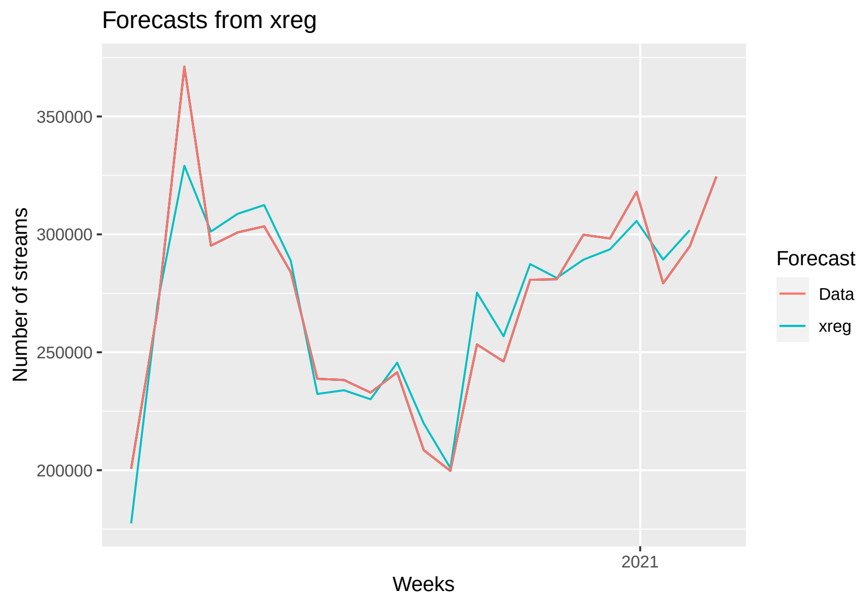


The following example demonstrates the effectiveness of dynamic regression with xreg= Rank, also the model passes the residuals check



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,0,0) errors
## Q* = 4.5644, df = 3, p-value = 0.2066
##
## Model df: 3.   Total lags used: 6
```

```
##                         ME         RMSE       MAE       MPE      MAPE MASE ACF1
## Training set        114.28     13156.97      9562     -0.09      3.51  NaN 0.04
## Test set      454081389.79 454081389.79 454081390 139940.83 139940.83  NaN   NA
```



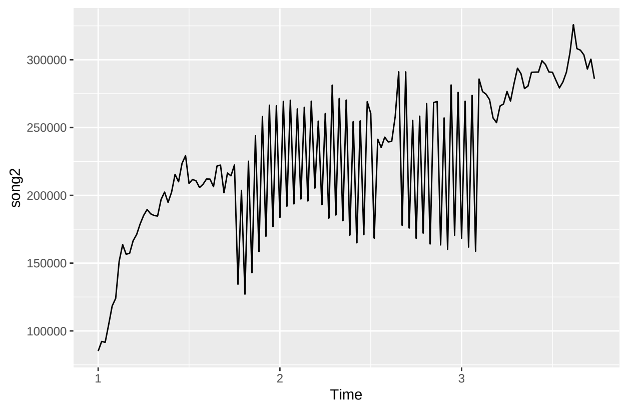
Pipeline for the Streams forecast

The Streamfcast_pipeline.R run the selected models on each song and choose the best one based on RMSE. Returns an Excel file as below:

```
## New names:
## * `` -> ...6
```

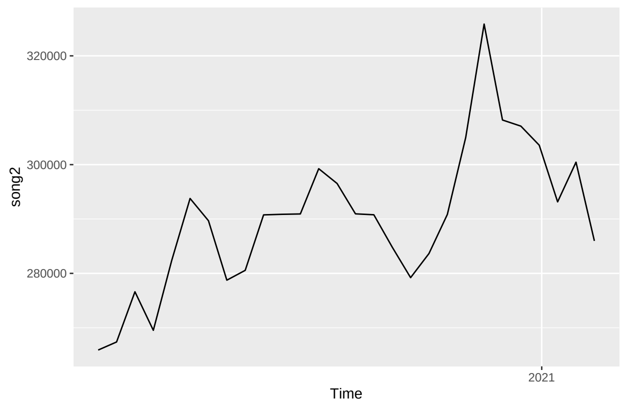
```
## # A tibble: 6 x 6
##   song                    Artist           RMSE Fcast   Model                ...6
##   <chr>                   <chr>           <dbl> <chr>   <chr>               <dbl>
## 1 Ver O Zaman Gömlekleri~ Sila, Yalin      100  490288  <NA>                   NA
## 2 Vazgeçtim Inan          Irem Derici      100  413697  <NA>                   NA
## 3 Sik                     Tepki              0  0       0                      NA
## 4 Whoopty                 CJ              35.4 693951~ Holt's method          NA
## 5 Bir Derdim Var          mor ve ötesi   1069.  276696~ Damped Holt's met~     NA
## 6 <NA>                    <NA>         919139.  <NA>    <NA>                   NA
```

Note: There has been some songs could not be handled by the pipeline mainly because they have very little data points(3 weeks) or because the *ts_ts()* failed to detect a pattern for the time-series due to duplicated entries. These sons(20 out of 200) will be handled manually, in addition to those that has high RMSE. For example this song shows no regular patter:



Therefore, it been processed manually by only considering data point after the irregular pattern and the next point is foretasted using naive method with taking the trend into consideration $Y_{t+1} = Y_t + (Y_t - Y_{t-1}$

```
## [time]: 'Date' [value]: 'Streams'
```


song2
Time