Follower Activity: Time Series

*In this document, I build a time series model to predict future follower activity on TikTok. The data was exported from my personal TikTok as a csv and can provide insight about optimal post-time and long-term follower activity behavior.*

*The data shows follower activity at each hour, over the course of a week. I decided to build with a SARIMA model because of its ability to handle complex seasonal patterns (in this case, a season being 24 hours).*

## Import Relevant Libraries

library(tidyverse)  
library(babynames)  
library(astsa)  
library(forecast)

## Read the data in

*Printing data to ensure data is read in correctly, chopped to 5 rows for the purpose of this markdown.*

fActivity <- read.csv("Follower activity.csv")  
head(fActivity,5)

## Date Hour Active.followers  
## 1 2022-11-18 1 4560  
## 2 2022-11-18 2 3666  
## 3 2022-11-18 3 2892  
## 4 2022-11-18 4 2314  
## 5 2022-11-18 5 1943

## Create a 1d vector with just follower activity

*The time series function being used expects 1d inputs.*

jFollowers<- fActivity %>%   
 select(Active.followers)  
  
head(jFollowers,5)

## Active.followers  
## 1 4560  
## 2 3666  
## 3 2892  
## 4 2314  
## 5 1943

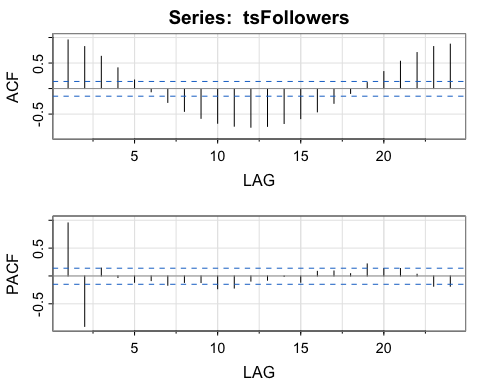
## Convert to Time Series

tsFollowers<-ts(jFollowers)

## Evaluate ACF and PACF

*Visually evaluating where threshold cutoffs for lag exist in ACF and PACF plots can indicate whether an auto regressive (AR), or a moving average (MA) model is a better fit. Since the PACF cuts off more significantly, this indicates that I should expect a more AR based model to be more applicable here.*

acf2(tsFollowers)



## See what model is suggested using an autofit function

*While this will likely not be my final model, it can serve as a good starting point for the model build.*

auto.arima(tsFollowers,trace=T)

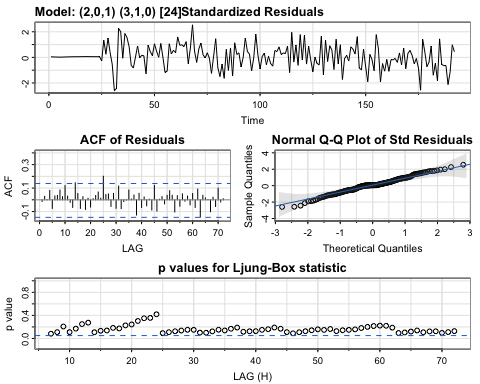
##   
## Fitting models using approximations to speed things up...  
##   
## ARIMA(2,0,2) with non-zero mean : 2495.514  
## ARIMA(0,0,0) with non-zero mean : 3327.936  
## ARIMA(1,0,0) with non-zero mean : 2873.573  
## ARIMA(0,0,1) with non-zero mean : 3085.218  
## ARIMA(0,0,0) with zero mean : 3805.061  
## ARIMA(1,0,2) with non-zero mean : 2612.54  
## ARIMA(2,0,1) with non-zero mean : 2494.992  
## ARIMA(1,0,1) with non-zero mean : 2707.064  
## ARIMA(2,0,0) with non-zero mean : 2501.491  
## ARIMA(3,0,1) with non-zero mean : 2497.325  
## ARIMA(3,0,0) with non-zero mean : 2495.266  
## ARIMA(3,0,2) with non-zero mean : 2496.118  
## ARIMA(2,0,1) with zero mean : 2553.864  
##   
## Now re-fitting the best model(s) without approximations...  
##   
## ARIMA(2,0,1) with non-zero mean : 2503.923  
##   
## Best model: ARIMA(2,0,1) with non-zero mean

## Series: tsFollowers   
## ARIMA(2,0,1) with non-zero mean   
##   
## Coefficients:  
## ar1 ar2 ma1 mean  
## 1.8052 -0.8986 0.2141 4678.3617  
## s.e. 0.0329 0.0327 0.0685 147.5478  
##   
## sigma^2 = 25270: log likelihood = -1246.8  
## AIC=2503.6 AICc=2503.92 BIC=2519.89

## Find Optimal Model by Comparing Residuals and Term Significance

*This model was selected by starting with the autofit suggestion and introducing a 24 hour seasonality term. From there I added terms, seeing the effect on the residual plots and whether each term was statistically significant. While it was impossible to find a model that reduced the p-values for the Ljung-Box statistic to under 0.05 by introducing only statistically significant terms. The following model produced expected ACF of residual plots and normal Q-Q plots to suggest that this would still be an effective model. Note the MA1 term that was introduced does fall a bit above a 0.05 threshold for significance, it is still a low p-value and seemed to visually improve the residuals to such an extent that I decided it was a valuable term to keep in the final model.*

sarima(tsFollowers,2,0,1,3,1,0,24)

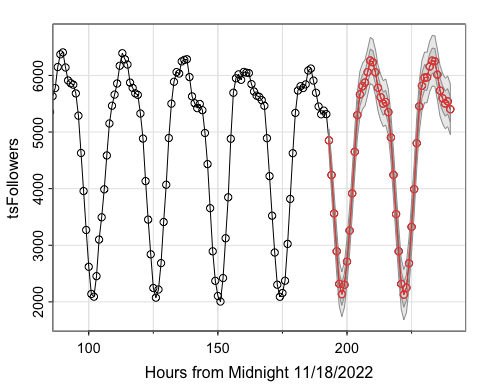


## $fit  
##   
## Call:  
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),   
## xreg = constant, transform.pars = trans, fixed = fixed, optim.control = list(trace = trc,   
## REPORT = 1, reltol = tol))  
##   
## Coefficients:  
## ar1 ar2 ma1 sar1 sar2 sar3 constant  
## 1.3788 -0.622 -0.1916 -0.5480 -0.5417 -0.4126 0.3612  
## s.e. 0.1096 0.095 0.1328 0.0874 0.0818 0.0899 0.5171  
##   
## sigma^2 estimated as 9893: log likelihood = -1023.49, aic = 2062.98  
##   
## $degrees\_of\_freedom  
## [1] 161  
##   
## $ttable  
## Estimate SE t.value p.value  
## ar1 1.3788 0.1096 12.5754 0.0000  
## ar2 -0.6220 0.0950 -6.5504 0.0000  
## ma1 -0.1916 0.1328 -1.4431 0.1509  
## sar1 -0.5480 0.0874 -6.2667 0.0000  
## sar2 -0.5417 0.0818 -6.6254 0.0000  
## sar3 -0.4126 0.0899 -4.5877 0.0000  
## constant 0.3612 0.5171 0.6985 0.4859  
##   
## $AIC  
## [1] 12.27962  
##   
## $AICc  
## [1] 12.28379  
##   
## $BIC  
## [1] 12.42838

## Plotting results from the model build

tsFollowers

forecasts<- sarima.for(tsFollowers,48,2,0,1,3,1,0,24, xlab = "Hours from Midnight 11/18/2022")



## Extracting the Data from the Forecast Function

*Extracting the forecast and the standard errors to create a confidence interval in a new plot.*

# Extract forecasted values and standard errors  
forecasted\_values <- forecasts$pred  
standard\_errors <- forecasts$se  
  
# Set the confidence level  
confidence\_level <- 0.95  
  
# Calculate the Z-score for the desired confidence level  
z\_score <- qnorm((1 + confidence\_level) / 2)  
  
# Calculate the margin of error  
margin\_of\_error <- z\_score \* standard\_errors  
  
# Calculate the lower and upper confidence intervals  
lower\_ci <- forecasted\_values - margin\_of\_error  
upper\_ci <- forecasted\_values + margin\_of\_error

*Formatting the data more nicely so that it’s easier to work with and introducing a date axis for the data.*

# Create a data frame for plotting  
forecast\_data <- data.frame(  
 Time = time(forecasted\_values),  
 Forecast = forecasted\_values,  
 Lower\_CI = lower\_ci,  
 Upper\_CI = upper\_ci  
)  
  
#set how many data points before prediction shows up on plot  
preShow=72  
  
#set x-axis dates  
xDates<-seq(as.POSIXct("2022-11-18 23:00:00"), as.POSIXct("2022-12-31 08:32:00"), by="hour")  
  
#select the dates for the data to be displayed  
dataDates<-xDates[seq(1:preShow)+length(tsFollowers)-preShow+1]  
  
#select the dates for the forecast  
forecastDates<-xDates[forecast\_data$Time]+1440

## Replotting

*For most purposes, the plot above should be sufficient. I decided to extract the data and replot in a prettier way to demonstrate data visualization skill.*

# Create a plot with forecasts and confidence intervals  
ggplot() +  
 #forecast  
 geom\_line(aes(x= c(tail(dataDates,1),forecastDates), y = c(tail(tsFollowers,1),forecast\_data$Forecast), color = "blue"), size = 0.5, linetype='dashed') +  
 geom\_point(aes(x= forecastDates, y = forecast\_data$Forecast, color = "blue"), size = 1) +  
 geom\_ribbon(aes(x = forecastDates, ymin = forecast\_data$Lower\_CI, ymax = forecast\_data$Upper\_CI, fill = "blue"), alpha = 0.2) +  
 #data  
 geom\_line(aes(x=dataDates,y=tail(tsFollowers,preShow),color="black")) +  
 geom\_point(aes(x=dataDates,y=tail(tsFollowers,preShow))) +  
 #legends, axes, theme, etc.  
 labs(x = "Date & Time", y = "Number of Active Followers", title= "Active Followers by Time") +  
 scale\_x\_datetime(labels = scales::date\_format("%m-%d %H:%M%p"), breaks = "1 day") +  
 scale\_color\_manual(name = "Color", values = c("black","blue"), labels = c("Measured data","Forecast")) +  
 scale\_fill\_manual(name = "Fill", values = c("blue"), labels = c("95% Confidence")) +  
 theme\_bw()

A graph showing the growth of a flower

Description automatically generated with medium confidence

*This plot can be referenced and recreated over different dates to help me determine optimal post time. I learned, for example, to expect most follower activity to occur in the afternoon.*