An Time Series Analysis and Forecasting of Daily Count of Rental Bikes in Capital Bike-sharing System

Time Series and Forecasting Final Project

Sijie Xiang | MA 585 | May 1, 2018

Introduction

Time series analysis can be used in a multitude of business applications for forecasting a quantity into the future and explaining its historical patterns. A few examples of possible use cases range from explaining seasonal patterns in sales to predicting the expected number of incoming airplane passengers. In this report, we are interested in real word applications of bike sharing systems as the characteristics of data being generated by these systems make them attractive for both research and application. For this project, our goal is to explore bike sharing dataset (the link in provided below) and identify the optimal time series model for future daily rental bike checkouts forecast.

In this analysis, our approach to the goal is twofold. In the very first approach I selected model myself. I plotted, examined, and prepared series for modeling, extracted the seasonality component from the time series, tested for stationary and applied appropriate transformation, chose the order of an ARIMA model and forecasted the series. Another approach is to use auto.arima() by brute force. In the end, I used various measures of forecast accuracy such as MAE, RMSE, and MAPE for model comparisons and eventually concluded my final model.

My conclusion is that the best quality of fit time series model is ARIMA (1,1,7).

Dataset Description

Descriptions of my dataset, as well as its key attributes, are as follows:

My dataset, gathered from the UCI Machina Learning Lab, indicates the hourly and daily count of rental bikes between years 2011 and 2012 in Capital bike-sharing system with the corresponding weather and seasonal information. Information is available in the following link.

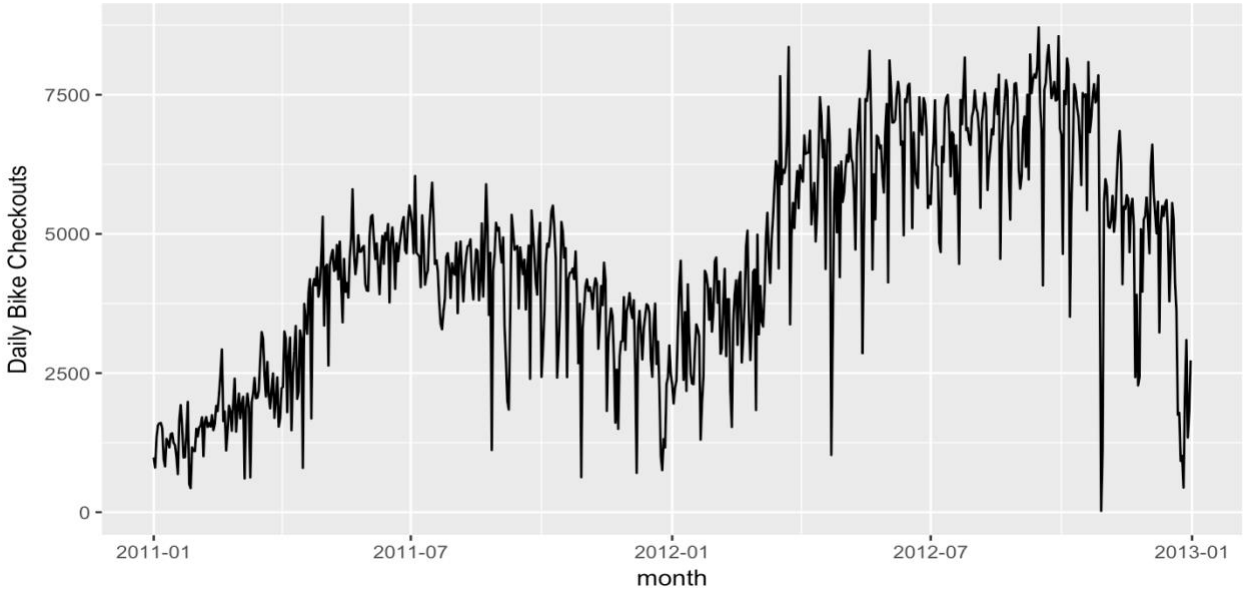
https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset#

Attribute	Type	Description
instant	Nominal	Record index
dteday	Nominal	Date
season	Nominal	Season(1-4)
yr	Nominal	Year(0:2011, 1:2022)
mnth	Nominal	Month(1 to 12)
holiday	Nominal	Whether day is holiday or not
weekday	Nominal	Day of the week
workingday	Nominal	If day is neither weekends nor holiday is 1, otherwise is 0
weathersit	Nominal	1: Clear, Partly cloudy 2: Mist + Cloudy 3: Light Snow, Light Rain
temp	Numeric	Normalized temperature in Celsius
atemp	Numeric	Normalized feeling temperature in Celsius
hum	Numeric	Normalized humidity
windspeed	Numeric	Normalized wind speed
casual	Numeric	Count of casual users
registered	Numeric	Count of registered users
cnt	Numeric	Count of total rental bikes including both casual and registered

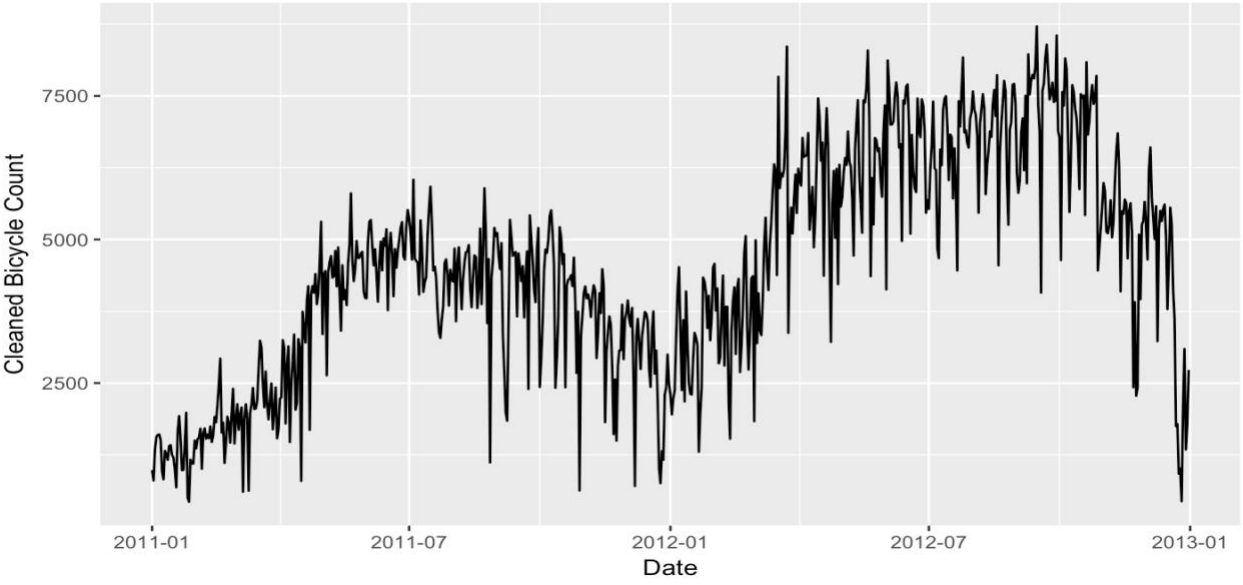
Data Preparation

This dataset that UCI Machine Learning Lab provides, consists of a number of information that I do not need to use. A lot of the attributes are irrelevant and not necessary for my research. Again, our goal is to build a time series model to forecast cnt (Count of total rental bikes including both casual and registered) at a given time and therefore I deleted all other attributes except cnt and dteday for data cleaning purposes. After checking there are no missing

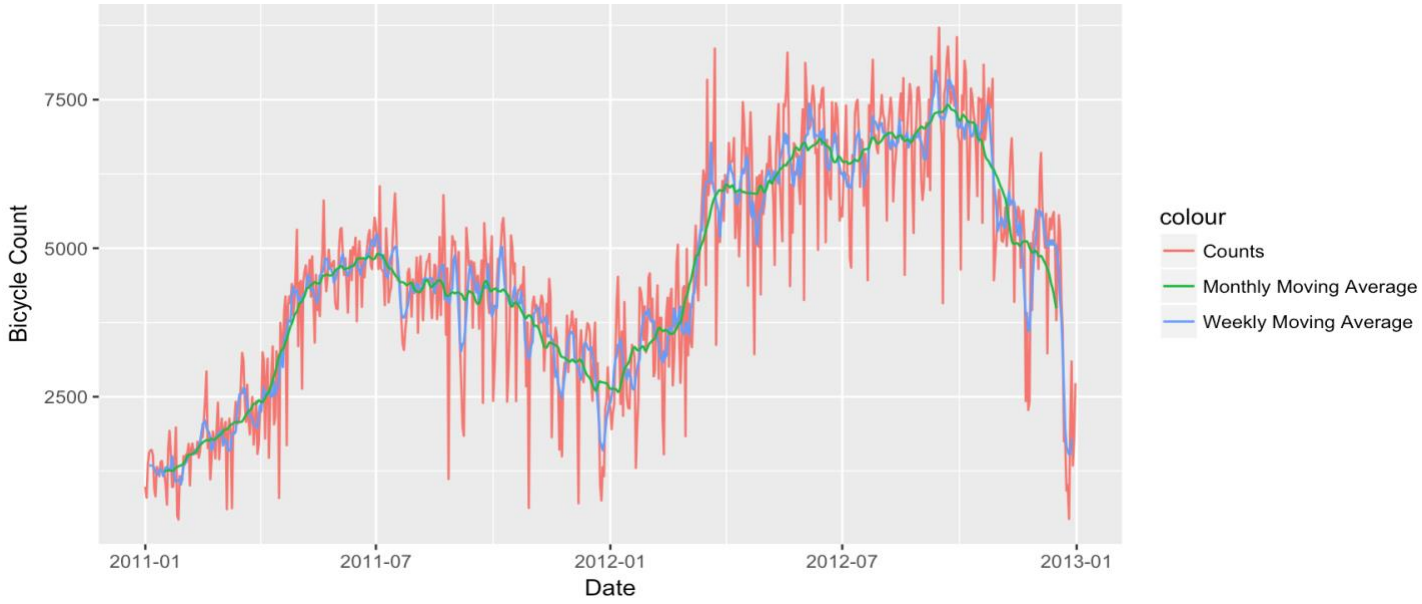
values in the dataset, I plotted the cnt series and visually examined it for any outliers, volatilities, and irregularities.



I observed that in some cases, the number of bicycles checkout dropped below 50 on day and rose to over 5,000 the next day. These are suspected outliers that could bias the model by skewing statistical summaries. Therefore, I decided to use ts.clean() to remove outliers and replotted the clean series using ggplot. Now our series contains fewer bad leverage points.



However, even after removing outliers, the daily data is still pretty volatile as volatile clustering occurred, which motivated me to use moving average to smooth the series tracing its bigger troughs and peaks while smoothing out noisy fluctuations. The wider the window of the moving average, the smoother original series becomes. In our bike sharing example, I take weekly and monthly moving average, smoothing the series into something more stable and therefore predictable.

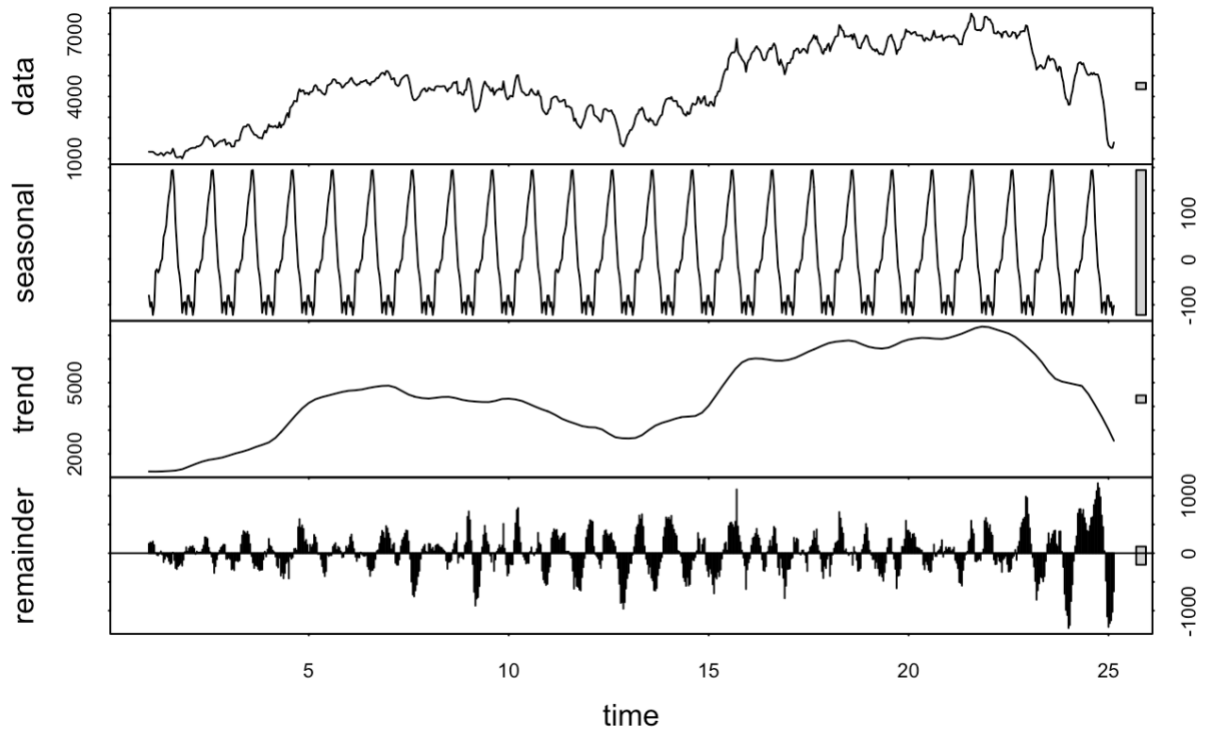


I observed that there is a seasonal component in out plot as lower usages of bicycles occur in the winter months and higher checkout numbers are observed in the summer month. Moreover, the amplitude of seasonal component seems to increase with level of the series. In this case, a multiplicative model might be more appropriate when I decompose my data.

Time Series Analysis and Results

As mentioned in my introduction, ARIMA models can be fitted to both seasonal and non-seasonal data. Therefore, I will first de-seasonalize the series and fit a "Vanilla" non-seasonal ARIMA model and later on construct a seasonal ARIMA model for comparisons. First, I

calculated seasonal component of the data and adjusted the original series by subtracting seasonality. We now have a de-seasonalized series and can proceed to the stationarity test.

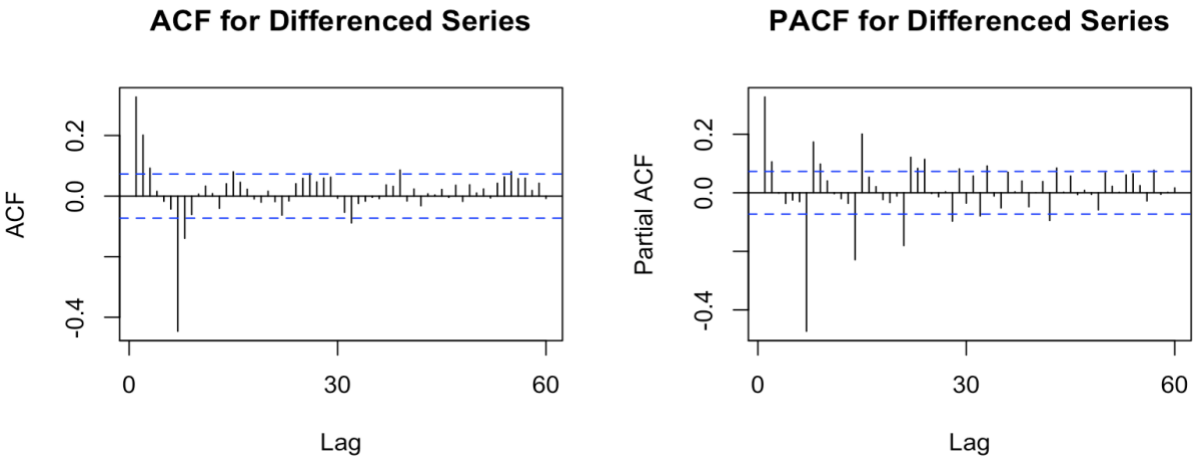


Fitting an ARIMA model requires a series to be stationary. A series is said to be stationary when its mean, variance, and auto-covariance are independent of time T. Through my visual inspection, I estimated our bicycle data is non-stationary because of trend component. In order to confirm my estimation, I employed augmented Dickey- Fuller(ADF) test for stationarity. The null hypothesis assumes that the series is non-stationary. Since p value is greater than 5%, I failed to reject the null hypothesis and concluded non-stationary.

```
        Augmented Dickey-Fuller Test

data:  count_ma
Dickey-Fuller = -0.2557, Lag order = 8, p-value = 0.99
alternative hypothesis: stationary
```

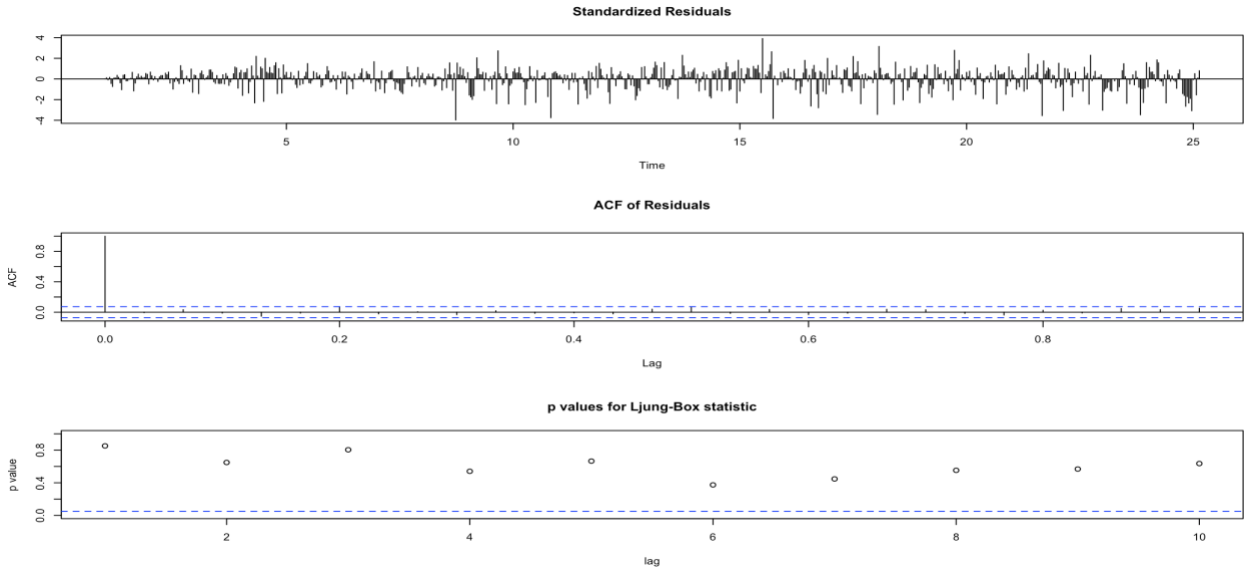
As we have discussed this in the class, nonstationarity can be removed via differencing. Although differencing the series can help in removing its trend or cycles, overdifferencing is undesirable. Hence, I started with the order of d = 1 and reevaluate whether further differencing is needed. The ADF test on differenced data rejected the null hypothesis of non-stationarity. Next, I calculated ACF and PACF of the differenced series to help inform the choice of p or q for my model.



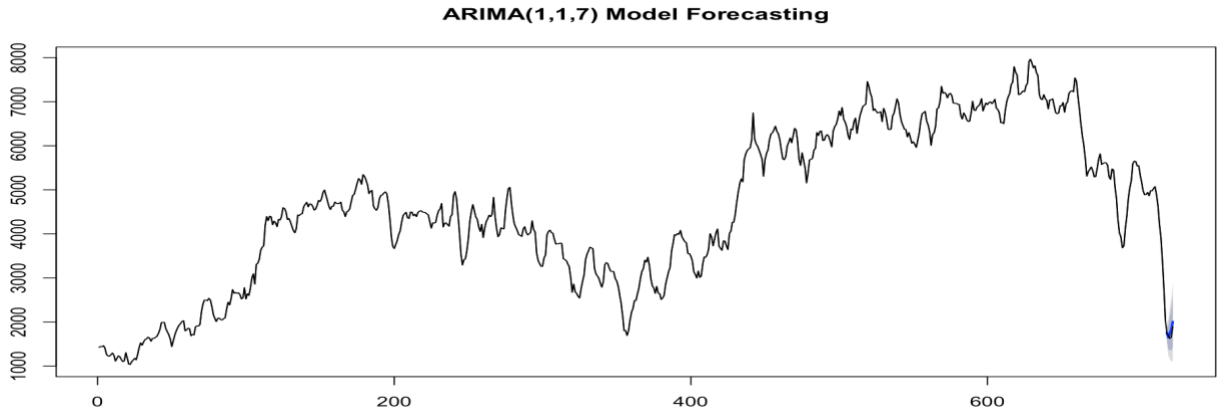
Plots show that there are strong autocorrelations at lag 1 and lag 2 and beyond. As both ACF and PACF slowly decay to zero, choosing a lower order ARIMA model such as ARIMA (1,1,1) seems appropriate. However, I also noticed that there is a clear pattern presented in both ACF and PACF at lag 7. This suggests that my model may be better off with different specification such as p = 7 or q = 7. To decide quality of fit across three non-seasonal ARIMA models, I carried out model comparisons test using Akaike information criteria (AIC). The table below suggested that ARIMA (1,1,7) has the best quality of fit among three models as it has the smallest AIC value.

	ARIMA(1,1,1)	ARIMA(1,1,7)	ARIMA(7,1,1)
AIC	9423.82	9024.56	9226.47

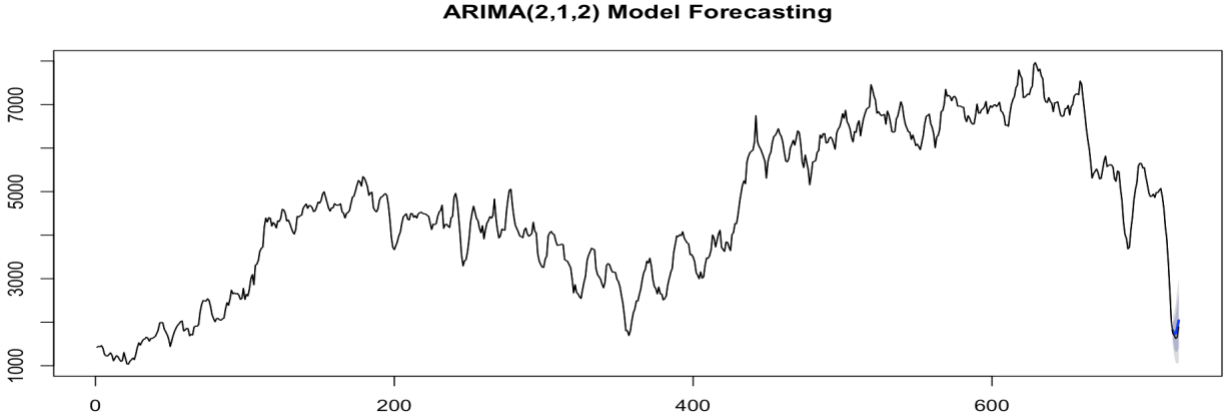
Once I have a candidate model, one important aspect if time series modeling involves techniques used to verify adequacy of the fitted model. Hypothesis testing procedures such as Ljung-box test (a modification of Portmanteau test) can be utilized for this purpose.



Based on the model diagnostics, I concluded that the residuals are realizations from a white noise process because ACF at each lag, where lag is greater than 0, falls inside the interval. In addition, p value at lag is greater than 5%, suggesting favor of null hypothesis of white noise. Now I wanted to get a sense of how the model will perform in the future. I would like to use out-of-sample forecast validation to address this problem. Normally, the size of validation sample is 15 – 20% of the total sample size. This suggested the forecast horizon (testing data) would last about 3 months. However, this rule of thumb is not appropriate here given the fact that Bicycle checkout counts drop and spike tremendously in Winter and Summer respectively and forecast horizon of three month is so long that total rental bikes checkout counts are very likely to suffer from regime change, which is disastrous to any time series forecasting model. Therefore, I kept forecast horizon short like 5 days.



With a short forecast horizon, my forecast, represented in blue line almost overlapped on top of the actual testing data. This a good sign because our forecast successfully picked up the trend even through a regime change is presented. While I will use various measures of forecast accuracy such as MAE, RMSE, and MAPE for model comparisons in the conclusion part, analysis above completed my first approach to the problem. Next, I used auto.arima() function directly and identified the best quality of fit is a ARIMA(2,1,2) model with a seasonal component taken into consideration. I went through the same procedure and plotted ARIMA (2,1,2) model 5 days forecast in the following. Now we have two competing models, it would be a great idea to conclude the final model via various forecast accuracy measures.



The table below recorded the performance of two competing models in the testing set. It suggested that ARIMA (1,1,7) would be a better model as it outperformed ARIMA (2,1,2) in any forecast measures that I provided.

	MAE	RMSE	MAPE
ARIMA(1,1,7)	92.96	5.45	109.14
ARIMA(2,1,2)	115.44	6.73	131.91

Conclusions

To reiterate, the purpose of this report is to identify an accurate and robust time series model that is able to predict daily rental bike checkouts. I gave a detail description of all features and also showed critical steps to clean up the data such as missing value detections, bad leverage points treatment before I moved into model selection steps. Again, my justification of model selection steps is twofold. My first approach involved selecting model myself and second approach included using built-in function auto.arima(). Having two competing models, I carried out the forecasting via hold-out methods and concluded my final model as ARIMA (1,1,7) after various measures of forecast accuracy such as MAE, RMSE, and MAPE for model comparisons in the result section. In the context of problem, my forecast indicated there would be an increasing demand in rental bikes checkouts and therefore I recommended transport services to put more bikes out to satisfy public needs.

	t + 1	t + 2	t + 3	t + 4	t + 5
cnt	**2093**	**2457**	**2565**	**2530**	**2688**

References

https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset#

R. Dalinina, “Introduction to Time Series Forecasting with ARIMA” 2017

Appendix A: Data Preparation

```
library(readr)
library(ggplot2)
library(forecast)
library(tseries)

daily_data = read.csv('day.csv', header=TRUE, stringsAsFactors=FALSE)
# Checking missing data
daily_data[!complete.cases(daily_data),]
daily_data$Date = as.Date(daily_data$dteday)
ggplot(daily_data, aes(Date, cnt)) +
  geom_line() + scale_x_date('month')  + ylab("Daily Bike Checkouts") +
  xlab("")

# Replot the CNT Series
count_ts = ts(daily_data[, c('cnt')])
daily_data$clean_cnt = tsclean(count_ts)
ggplot() +
  geom_line(data = daily_data, aes(x = Date, y = clean_cnt)) +
  ylab('Cleaned Bicycle Count')

# Use Moving average to smooth CNT series
daily_data$cnt_ma = ma(daily_data$clean_cnt, order=7) # using the clean count with no outliers
daily_data$cnt_ma30 = ma(daily_data$clean_cnt, order=30)

ggplot() +
  geom_line(data = daily_data, aes(x = Date, y = clean_cnt, colour = "Counts")) +
  geom_line(data = daily_data, aes(x = Date, y = cnt_ma,   colour = "Weekly Moving Average"))  +
  geom_line(data = daily_data, aes(x = Date, y = cnt_ma30, colour = "Monthly Moving Average"))  +
  ylab('Bicycle Count')
```

Appendix B: Times Series Analysis and Results

```
# Remove Seasonal Component
count_ma = ts(na.omit(daily_data$cnt_ma), frequency=30)
decomp = stl(count_ma, s.window="periodic")
deseasonal_cnt <- seasadj(decomp)
plot(decomp)

# Stationarity Test
adf.test(count_ma)

# Apply differencing
count_d1 = diff(deseasonal_cnt,1)
adf.test(count_d1)

# reapply ADF and PADF test
Acf(count_d1, main='ACF for Differenced Series')
Pacf(count_d1, main='PACF for Differenced Series')

# Compare three non-seasonal ARIMA model using AIC
ARIMA_111 = arima(deseasonal_cnt, order = c(1,1,1))
ARIMA_111
ARIMA_117 = arima(deseasonal_cnt, order = c(1,1,7))
ARIMA_117
ARIMA_711 = arima(deseasonal_cnt, order = c(7,1,1))
ARIMA_711

# Run diagonositics
tsdiag(ARIMA_117)

# Forecasting with select model myself
train=deseasonal_cnt[1:720]
test=deseasonal_cnt[721:725]
fit_1=arima(train,order=c(1,1,7))
arimafcast_1=forecast(fit_1,h=5)
arimafcast_1$mean
arimaerr_1=test-arimafcast_1$mean
arimamae_1=mean(abs(arimaerr_1))
arimarmse_1=sqrt(mean(arimaerr_1^2))
arimamape_1=mean(abs((arimaerr_1*100)/test))
par(mfrow=c(1,1))
plot(arimafcast_1, main="ARIMA(1,1,7) Model Forecasting")
lines(ts(deseasonal_cnt))

# Forecasting with select model with auto_arima
fit_w_seasonality = auto.arima(train,seasonal = TRUE, max.p = 10, max.q = 10)
arimafcast_2=forecast(fit_w_seasonality,h=5)
arimafcast_2$mean
arimaerr_2=test-arimafcast_2$mean
arimamae_2=mean(abs(arimaerr_2))
arimarmse_2=sqrt(mean(arimaerr_2^2))
arimamape_2=mean(abs((arimaerr_2*100)/test))
plot(arimafcast_2, main="ARIMA(2,1,2) Model Forecasting")
lines(ts(deseasonal_cnt))

# Final Model Forecasting next 5 days
Final_forecast = forecast(arima(deseasonal_cnt,order=c(1,1,7)),h=5)
```