Bike Renting

Jinal Patel

11 March 2018

Contents

Chapter 1

Introduction

1.1 Problem Statement

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings.

These days bike is becoming popular as people are using it more and more for recreation. Also, bike riding is becoming part of a healthy lifestyle as well. Even the government promote bike ridership to reduce traffic.

Bike sharing systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Using these systems, people are able to rent a bike from a one location and return it to a different place on an as-needed basis. Currently, there are over 500 bike-sharing programs around the world.

Bike sharing systems therefore function as a sensor network, which can be used for studying mobility in a city.In this analysis we will analyse the dataset and create a predictive model to forecast future ridership. Dataset shows the daily rental activity with parameters such as weather, days, humidity, etc. from the year 2011 to 2012.

1.2 Data

Our task is to build regression models which will predict the count of Bikes rented based on Time series analysis. Given below is a sample of the data set that we are using to predict the count of bikes:

Table 1.1: Bikes Rented Sample Data (Columns: 1-6)

instant	dteday	season	yr	mnth	holiday
1	01-01-2011	1	0	1	0
2	02-01-2011	1	0	1	0
3	03-01-2011	1	0	1	0
4	04-01-2011	1	0	1	0
5	05-01-2011	1	0	1	0
6	06-01-2011	1	0	1	0

Table 1.2: Bikes Rented Sample Data (Columns: 7-12)

weekday	workingday	weathersit	temp	atemp	hum
6	0	2	0.344167	0.363625	0.805833
0	0	2	0.363478	0.353739	0.696087
1	1	1	0.196364	0.189405	0.437273
2	1	1	0.2	0.212122	0.590435
3	1	1	0.226957	0.22927	0.436957
4	1	1	0.204348	0.233209	0.518261

Table 1.3: Bikes Rented Sample Data (Columns: 13-16)

windspeed	casual	registered	cnt
0.160446	331	654	985
0.248539	131	670	801
0.248309	120	1229	1349
0.160296	108	1454	1562
0.1869	82	1518	1600
0.0895652	88	1518	1606

As you can see in the table below we have the following 15 variables, using which we have to correctly predict the count of the Bikes rented:

Table 1.4: Predictor Variables

S.No.	Predictor
1	instant
2	dteday
3	season
4	yr
5	mnth
6	holiday
7	weekday
8	workingday
9	weathersit
10	temp
11	atemp
12	hum
13	windspeed

14	casual
15	Registered

Chapter 2

Methodology

2.1 Pre Processing

Any predictive modelling requires that we look at the data before we start modelling. So the first step towards data analysis is to explore the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as **Exploratory Data Analysis**. To start this process we will first try and look at all the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed. We can visualize that in a glance by looking at the probability distributions or probability density functions of the variable.

Initially we convert the variables to their appropriate form so that it is easy to process further. Type conversions is necessary for successful model development.

We see that the variable dteday is the Date on which the data is collected. It should be converted to Date datatype so that R and Python could differentiate between other variables and Date.

Other conversions include converting season (1:springer, 2:summer, 3:fall, 4:winter), yr(0: 2011, 1:2012), mnth(1 to 12), holiday(whether day is holiday or not (extracted from Holiday Schedule)), weekday(Day of the week), workingday(If day is neither weekend nor holiday is 1, otherwise is 0) and weathersit(1: Clear, Few clouds, Partly cloudy, Partly cloudy , 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist , 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds , 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog) from numerical to factor. We do this since they are discrete.

Likewise the remaining variables like instant, casual, registered and cnt should be converted to numeric.

2.1.1 Missing Value Analysis

This is an important step in Exploratory Data Analysis since we can’t feed missing values to our model. We calculate the percentage of all missing values in our dataset and then impute it appropriately with suitable imputation technique like Central statistics or else KNN imputation.

But the dataset we are working on doesn’t have any missing values or any spaces which needs to be replaced. Thus we can feed the data to the model.

2.2 Modelling

2.2.1 Model Selection

The dataset we are dealing with is a time series dataset. The evidence is its variation with time.

A time series is a series of data points indexed in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Thus it is a sequence of discrete-time data.

Models for time series data can have many forms and represent different stochastic processes. When modelling variations in the level of a process, three broad classes of practical importance are the autoregressive (AR) models, the integrated (I) models, and the moving average (MA) models. These three classes depend linearly on previous data points. Combinations of these ideas produce autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA) models.

ARIMA models are a popular and flexible class of forecasting model that utilize historical information to make predictions. This type of model is a basic forecasting technique that can be used as a foundation for more complex models.

We will go for ARIMA (autoregressive integrated moving average) method based on the evidence of a time series data.

2.2.2 ARIMA Model

ARIMA stands for auto-regressive integrated moving average and is specified by these three order parameters: (p, d, q). The process of fitting an ARIMA model is sometimes referred to as the Box-Jenkins method.

An auto regressive (AR(p)) component is referring to the use of past values in the regression equation for the series Y. The auto-regressive parameter p specifies the number of lags used in the model. For example, forecasting that if it rained a lot over the past few days, you state it’s likely that it will rain tomorrow as well. For example, AR(2) or, equivalently, ARIMA(2,0,0), is represented as

$$Y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + e_t$$

where ϕ1, ϕ2 are parameters for the model.

The d represents the degree of differencing in the integrated (I(d)) component. Differencing a series involves simply subtracting its current and previous values d times. You can imagine an example of this as forecasting that the amount of rain tomorrow will be similar to the amount of rain today, if the daily amounts of rain have been similar over the past few days. Often, differencing is used to stabilize the series when the stationarity assumption is not met, which we will discuss below.

A moving average (MA(q)) component represents the error of the model as a combination of previous error terms et. The order q determines the number of terms to include in the model.

$$Y_t = c + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \dots + \theta_q e_{t-q} + e_t$$

Differencing, autoregressive, and moving average components make up a non-seasonal ARIMA model which can be written as a linear equation:

$$Y_t = c + \phi_1 y_{d\ t-1} + \phi_p y_{d\ t-p} + \dots + \theta_1 e_{t-1} + \theta_q e_{t-q} + e_t$$

where yd is Y differenced d times and c is a constant.

First we will assumes the model to be non-seasonal series, which means you might need to de-seasonalize the series before modelling. We will show how this can be done in an example below.

ARIMA models can be also specified through a seasonal structure. In this case, the model is specified by two sets of order parameters: (p, d, q) as described above and $(P, D, Q)_m$ parameters describing the seasonal component of m periods.

ARIMA methodology does have its limitations. These models directly rely on past values, and therefore work best on long and stable series. Also note that ARIMA simply approximates historical patterns and therefore does not aim to explain the structure of the underlying data mechanism.

2.2.2 ARIMA Model Development

Step 1: Load Packages

We start out by loading the necessary packages and reading in the analysis dataset. Here we are using a dataset on the number of bicycles checkouts.

Step 2: Examine Your Data

A good starting point is to plot the series and visually examine it for any outliers, volatility, or irregularities. In this case, bicycle checkouts are showing a lot of fluctuations from one day to another. However, even with this volatility present, we already see some patterns emerge. For example, lower usage of bicycles occurs in the winter months and higher checkout numbers are observed in the summer months:

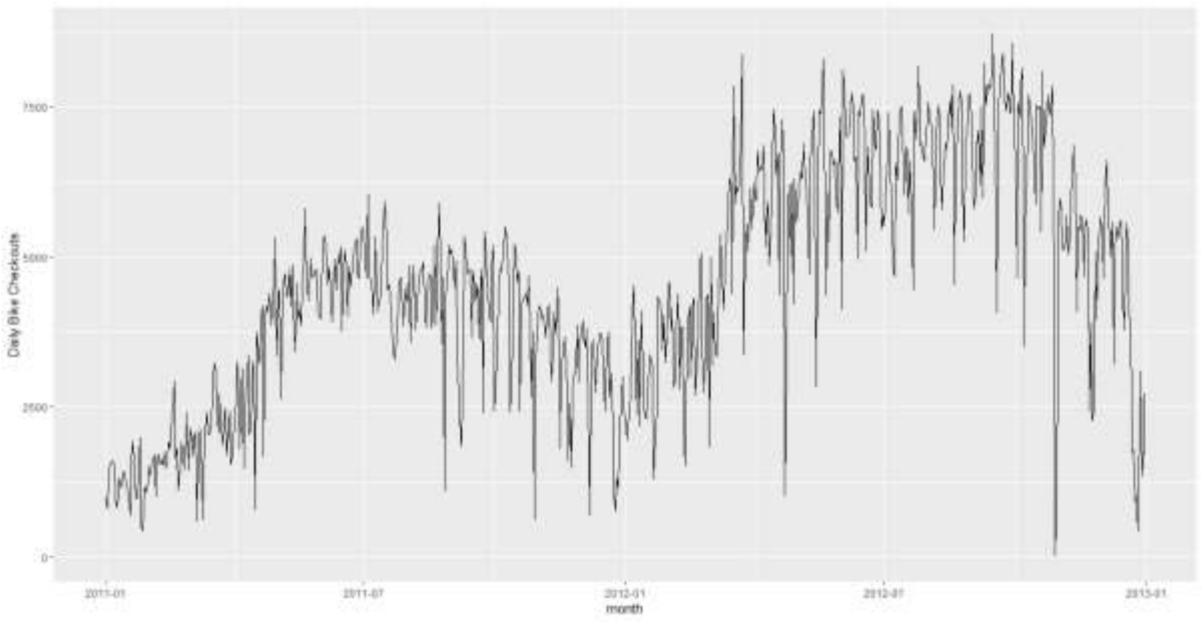


Figure 2.1: Distribution of Rental count

In some cases, the number of bicycles checked out dropped below 100 on day and rose to over 4,000 the next day. These are suspected outliers that could bias the model by skewing statistical summaries. R provides a convenient method for removing time series outliers: tsclean() as part of its forecast package. tsclean() identifies and replaces outliers using series smoothing and decomposition.

This method is also capable of inputing missing values in the series if there are any. Note that we are using the ts() command to create a time series object to pass to tsclean():

We plot the clean series using ggplot:

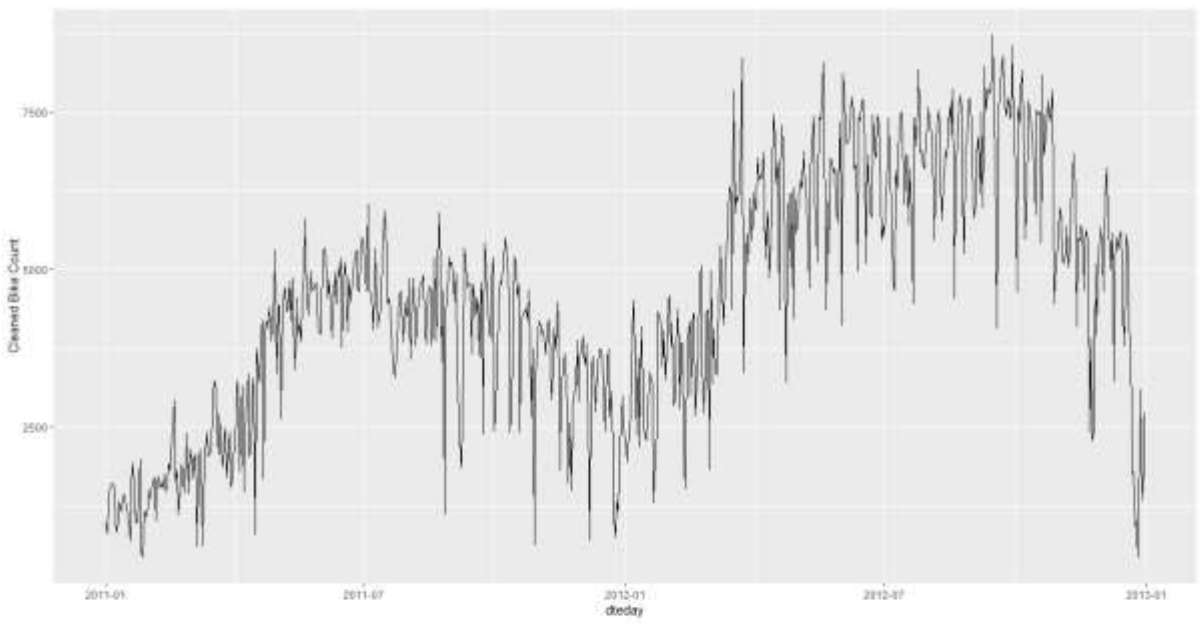


Figure 2.2: Distribution of Cleaned Rental count free from outliers

Even after removing outliers, the daily data is still pretty volatile. Visually, we could a draw a line through the series tracing its bigger troughs and peaks while smoothing out noisy fluctuations. This line can be described by one of the simplest — but also very useful —concepts in time series analysis known as a moving average. It is an intuitive concept that averages points across several time periods, thereby smoothing the observed data into a more stable predictable series.

Formally, a moving average (MA) of order m can be calculated by taking an average of series Y, k periods around each point:

$$MA = \frac{1}{m}\sum_{j=-k}^{k} y_{t+j}$$

where m = 2k + 1. The above quantity is also called a symmetric moving average because data on each side of a point is involved in the calculation.

Note that the moving average in this context is distinct from the M A(q) component in the above ARIMA definition. Moving average M A(q) as part of the ARIMA framework refers to error lags and combinations, whereas the summary statistic of moving average refers to a data smoothing technique.

The wider the window of the moving average, the smoother original series becomes. In our bicycle example, we can take weekly or monthly moving average, smoothing the series into something more stable and therefore predictable:

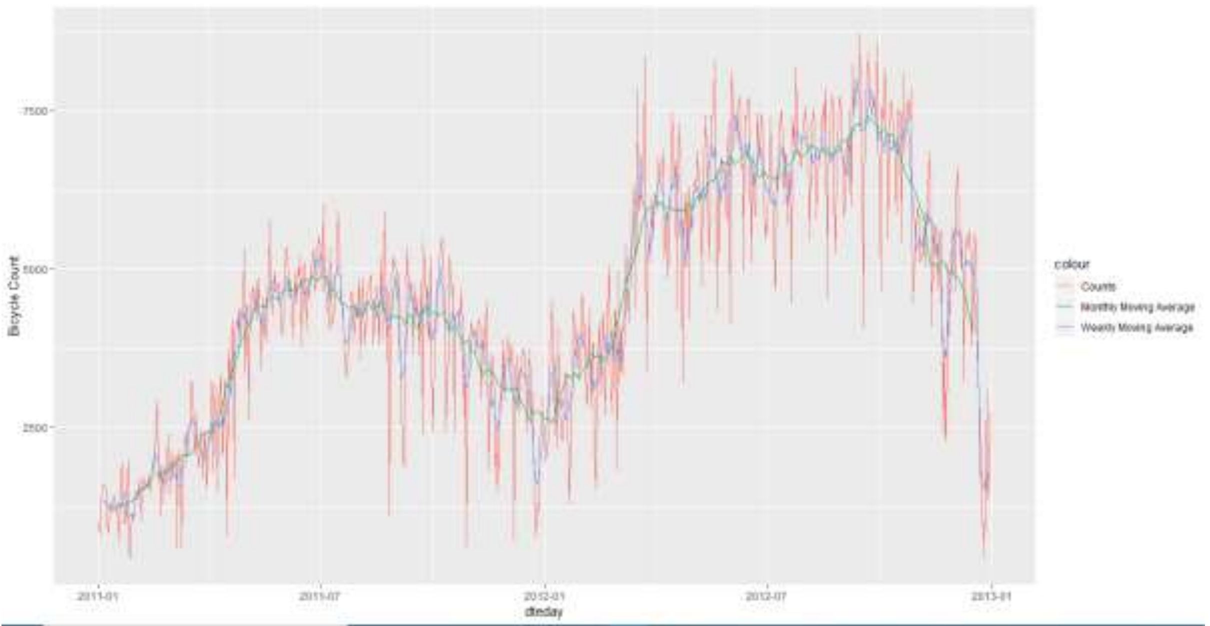


Figure 2.3: Bicycle count

In addition to volatility, modelling daily level data might require specifying multiple seasonality levels, such day of the week, week of the year, month of the year, holidays, etc. For the sake of simplicity, we will model the smoothed series of weekly moving average (as shown by the blue line above).

Step 3: Decompose Your Data

The building blocks of a time series analysis are seasonality, trend, and cycle. These intuitive components capture the historical patterns in the series. Not every series will have all three (or any) of these components, but if they are present, deconstructing the series can help you understand its behaviour and prepare a foundation for building a forecasting model.

Seasonal component refers to fluctuations in the data related to calendar cycles. For example, more people might be riding bikes in the summer and during warm weather, and less during colder months. Usually, seasonality is fixed at some number; for instance, quarter or month of the year.

Trend component is the overall pattern of the series: Is the number of bikes rented increasing or decreasing over time?

Cycle component consists of decreasing or increasing patterns that are not seasonal. Usually, trend and cycle components are grouped together. Trend-cycle component is estimated using moving averages.

Finally, part of the series that can't be attributed to seasonal, cycle, or trend components is referred to as **residual or error**.

The process of extracting these components is referred to as **decomposition**.

Formally, if Y is the number of bikes rented, we can decompose the series in two ways: by using either an additive or multiplicative model,

$$Y = S_t * T_t * E_t$$

where St is the seasonal component, T is trend and cycle, and E is the remaining error.

An additive model is usually more appropriate when the seasonal or trend component is not proportional to the level of the series, as we can just overlay (i.e. add) components together to reconstruct the series. On the other hand, if the seasonality component changes with the level or trend of the series, a simple "overlay," or addition of components, won't be sufficient to reconstruct the series. In that case, a multiplicative model might be more appropriate.

As mentioned above, ARIMA models can be fitted to both seasonal and non-seasonal data. Seasonal ARIMA requires a more complicated specification of the model structure, although the process of determining (P, D, Q) is similar to that of choosing non-seasonal order parameters. Therefore, we will explore how to de-seasonalize the series and use a "vanilla" non-seasonal ARIMA model.

First, we calculate seasonal component of the data using stl(). STL is a flexible function for decomposing and forecasting the series. It calculates the seasonal component of the series using smoothing, and adjusts the original series.

In the case of additive model structure, the same task of decomposing the series and removing the seasonality can be accomplished by simply subtracting the seasonal component from the original series. seasadj() is a convenient method inside the forecast package.

As for the frequency parameter in ts() object, we are specifying periodicity of the data, i.e., number of observations per period. Since we are using smoothed daily data, we have 30 observations per month.

We now have a de-seasonalized series and can proceed to the next step.

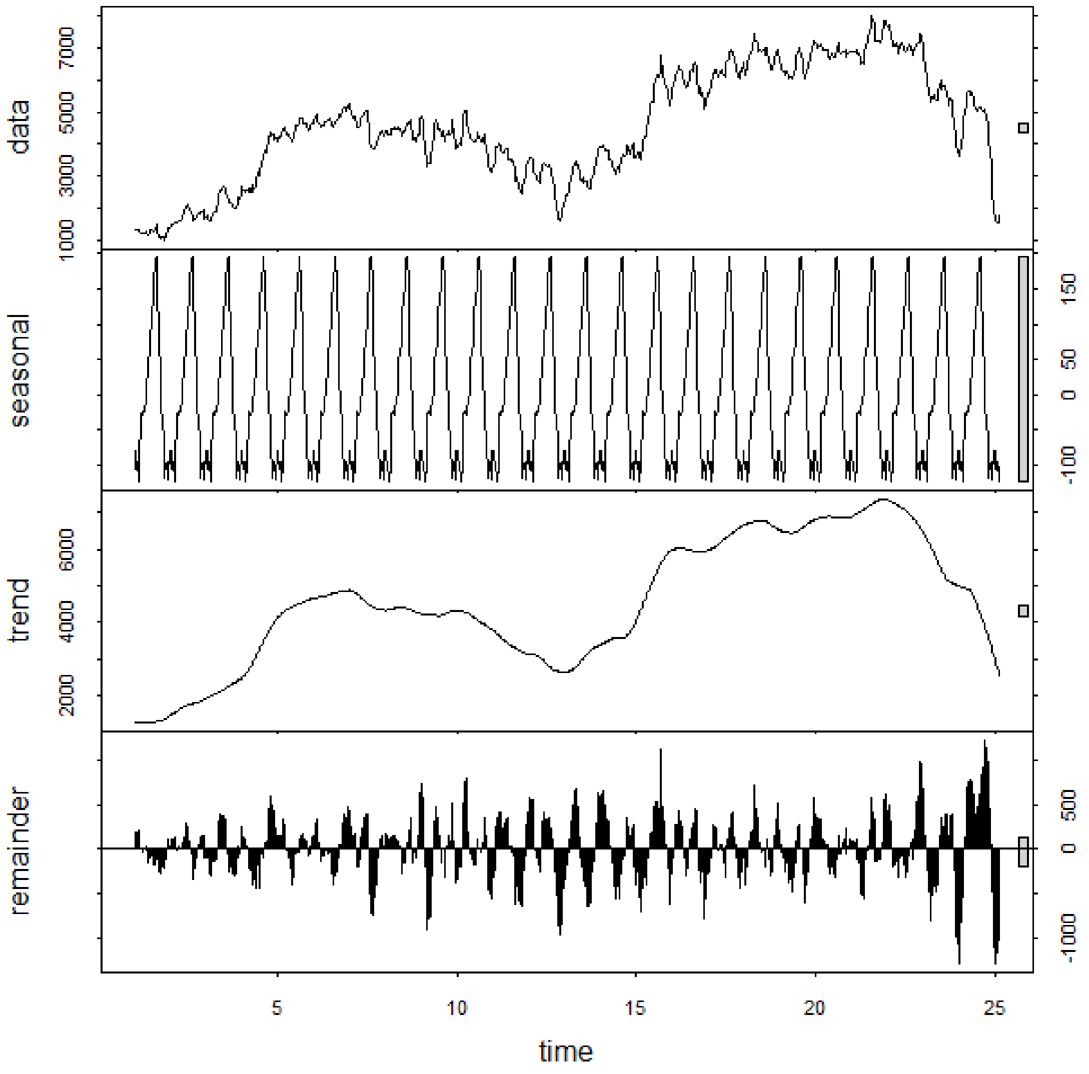
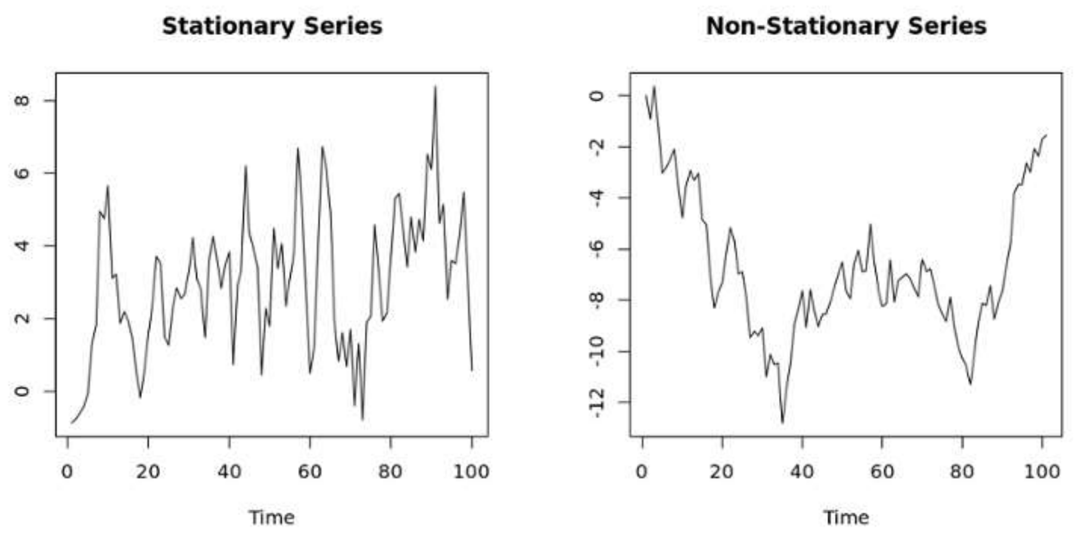


Figure 2.3: De-seasonalised Data

Step 4: Stationarity

Fitting an ARIMA model requires the series to be stationary. A series is said to be stationary when its mean, variance, and autocovariance are time invariant. This assumption makes intuitive sense: Since ARIMA uses previous lags of series to model its behavior, modeling stable series with consistent properties involves less uncertainty. The left panel below shows an example of a stationary series, where data values oscillate with a steady variance around the mean of 1. The panel on the right shows a non-stationary series; mean of this series will differ across different time windows.



The augmented Dickey-Fuller (ADF) test is a formal statistical test for stationarity. The null hypothesis assumes that the series is non-stationary. ADF procedure tests whether the change in Y can be explained by lagged value and a linear trend. If contribution of the lagged value to the change in Y is non-significant and there is a presence of a trend component, the series is non-stationary and null hypothesis will not be rejected.

Our bicycle data is non-stationary; the average number of bike checkouts changes through time, levels change, etc. A formal ADF test does not reject the null hypothesis of non-stationarity, confirming our visual inspection.

Usually, non-stationary series can be corrected by a simple transformation such as differencing. Differencing the series can help in removing its trend or cycles. The idea behind differencing is that, if the original data series does not have constant properties over time, then the change from one period to another might. The difference is calculated by subtracting one period's values from the previous period's values:

$$Y_{d_t} = Y_t - Y_{t-1}$$

Higher order differences are calculated in a similar fashion. For example, second order differencing (d = 2) is simply expanded to include second lag of the series:

$$Y_{d2_t} = Y_{d_t} - Y_{d_t - 1} = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2})$$

Similarly, differencing can be used if there is a seasonal pattern at specific lags. In such a case, subtracting a value for the "season" from a previous period represents the change from one period to another, as well as from one season to another:

$$Y_{d_t} = (Y_t - Y_{t-s}) - (Y_{t-1} - Y_{t-s-1})$$

The number of differences performed is represented by the d component of ARIMA. Now, we move on to diagnostics that can help determine the order of differencing.

Step 5: Autocorrelations and Choosing Model Order

Autocorrelation plots (also known as ACF or the auto correlation function) are a useful visual tool in determining whether a series is stationary. These plots can also help to choose the order parameters for ARIMA model. If the series is correlated with its lags then, generally, there are some trend or seasonal components and therefore its statistical properties are not constant over time.

ACF plots display correlation between a series and its lags. In addition to suggesting the order of differencing, ACF plots can help in determining the order of the M A (q) model. Partial autocorrelation plots (PACF), as the name suggests, display correlation between a variable and its lags that is not explained by previous lags. PACF plots are useful when determining the order of the AR(p) model.

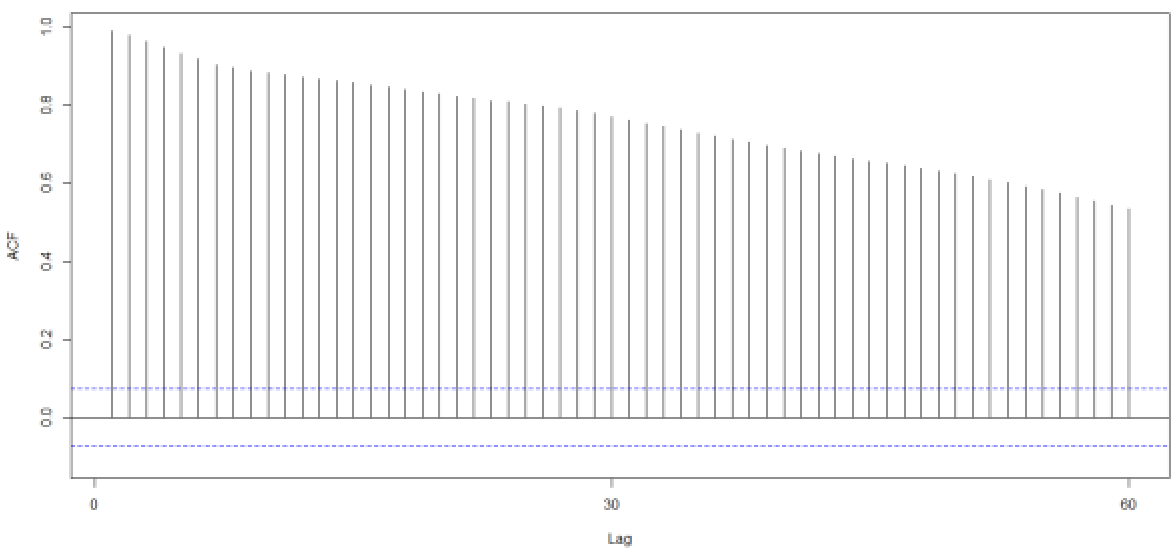


Figure 2.4: ACF plot

R plots 95% significance boundaries as blue dotted lines. There are significant autocorrelations with many lags in our bike series, as shown by the ACF plot below. However, this could be due to carry-over correlation from the first or early lags, since the PACF plot only shows a spike at lags 1 and 7:

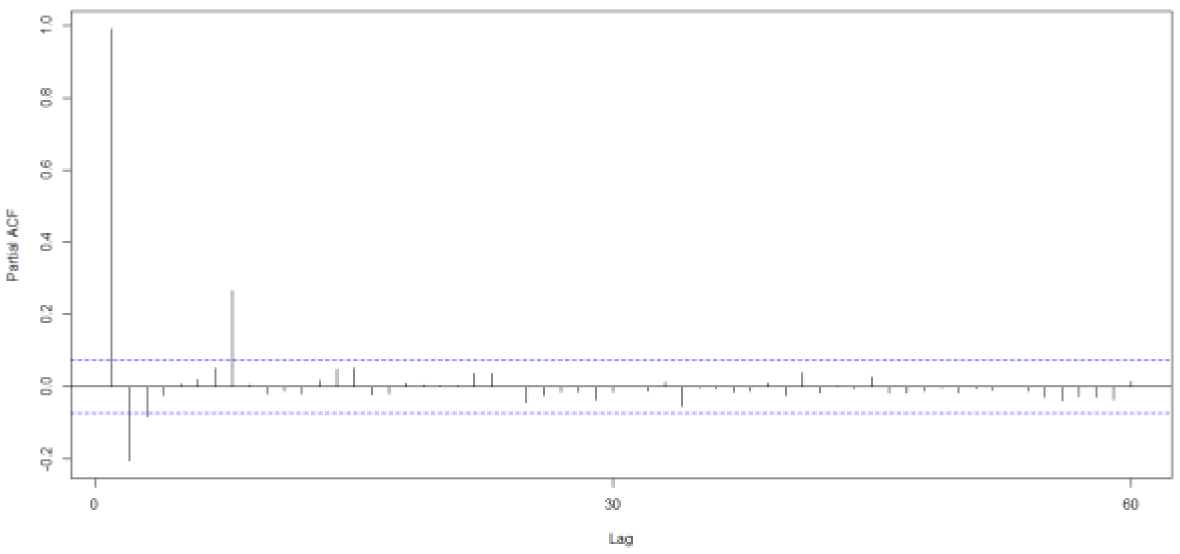


Figure 2.4: ACF plot

We can start with the order of d = 1 and re-evaluate whether further differencing is needed.

The augmented Dickey-Fuller test on differenced data rejects the null hypotheses of non-stationarity. Plotting the differenced series, we see an oscillating pattern around 0 with no visible strong trend. This suggests that differencing of order 1 terms is sufficient and should be included in the model.

Next, spikes at particular lags of the differenced series can help inform the choice of p or q for our model:

There are significant auto correlations at lag 1 and 2 and beyond. Partial correlation plots show a significant spike at lag 1 and 7. This suggests that we might want to test models with AR or MA components of order 1, 2, or 7. A spike at lag 7 might suggest that there is a seasonal pattern present, perhaps as day of the week. We talk about how to choose model order in the next step.

Step 6: Fitting an ARIMA model

Now let's fit a model. The forecast package allows the user to explicitly specify the order of the model using the arima() function, or automatically generate a set of optimal (p, d, q) using auto.arima(). This function searches through combinations of order parameters and picks the set that optimizes model fit criteria.

There exist a number of such criteria for comparing quality of fit across multiple models. Two of the most widely used are Akaike information criteria (AIC) and Baysian information criteria (BIC). These criteria are closely related and can be interpreted as an estimate of how much information would be lost if a given model is chosen. When comparing models, one wants to minimize AIC and BIC.

Step 7: Evaluate and Iterate

So now we have fitted a model that can produce a forecast, but does it make sense? Can we trust this model? We can start by examining ACF and PACF plots for model residuals. If model order parameters and structure are correctly specified, we would expect no significant autocorrelations present.

While auto.arima() can be very useful, it is still important to complete steps 1-5 in order to understand the series and interpret model results. Note that auto.arima() also allows the user to specify maximum order for (p, d, q), which is set to 5 by default.

We can specify non-seasonal ARIMA structure and fit the model to de-seasonalize data. Parameters (1,1,1) suggested by the automated procedure are in line with our expectations based on the steps above; the model incorporates differencing of degree 1, and uses an autoregressive term of first lag and a moving average model of order 1:

There is a clear pattern present in ACF/PACF and model residuals plots repeating at lag 7. This suggests that our model may be better off with a different specification, such as p = 7 or q = 7.

We can repeat the fitting process allowing for the MA(7) component and examine diagnostic plots again. This time, there are no significant autocorrelations present. If the model is not correctly specified, that will usually be reflected in residuals in the form of trends, skeweness, or any other patterns not captured by the model. Ideally, residuals should look like white noise, meaning they are normally distributed. A convenience function tsdisplay() can be used to plot these model diagnostics. Residuals plots show a smaller error range, more or less centered around 0. We can observe that AIC is smaller for the (1, 1, 7) structure as well:

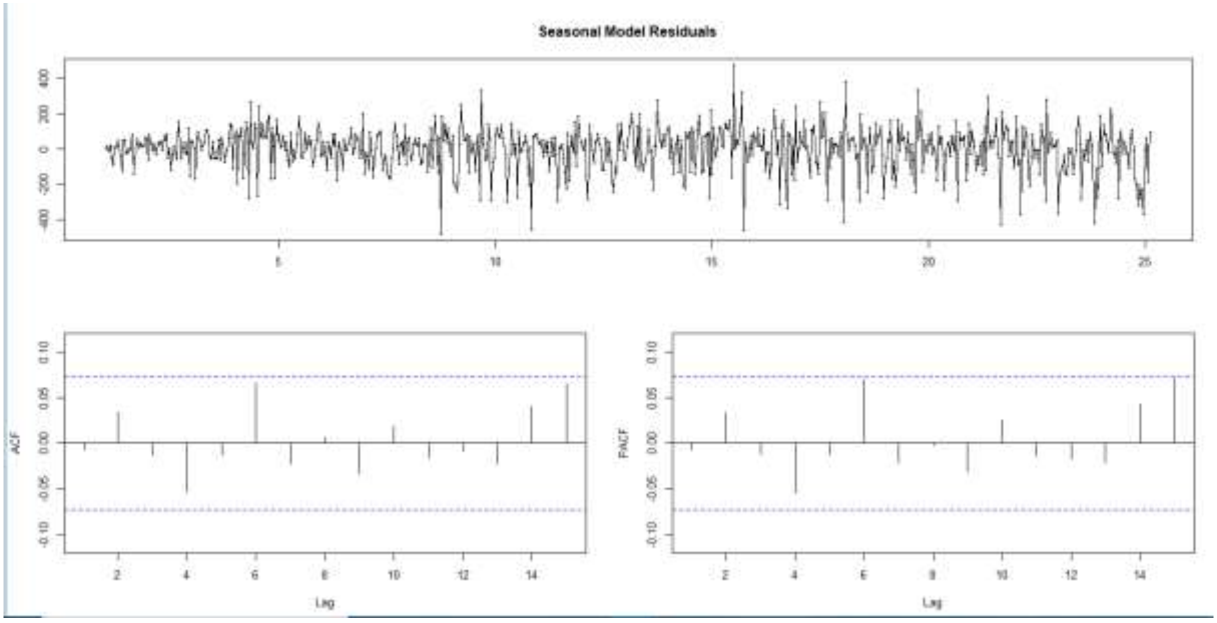


Figure: Plotting the seasonal model residuals

Forecasting using a fitted model is straightforward in R. We can specify forecast horizon h periods ahead for predictions to be made, and use the fitted model to generate those predictions:

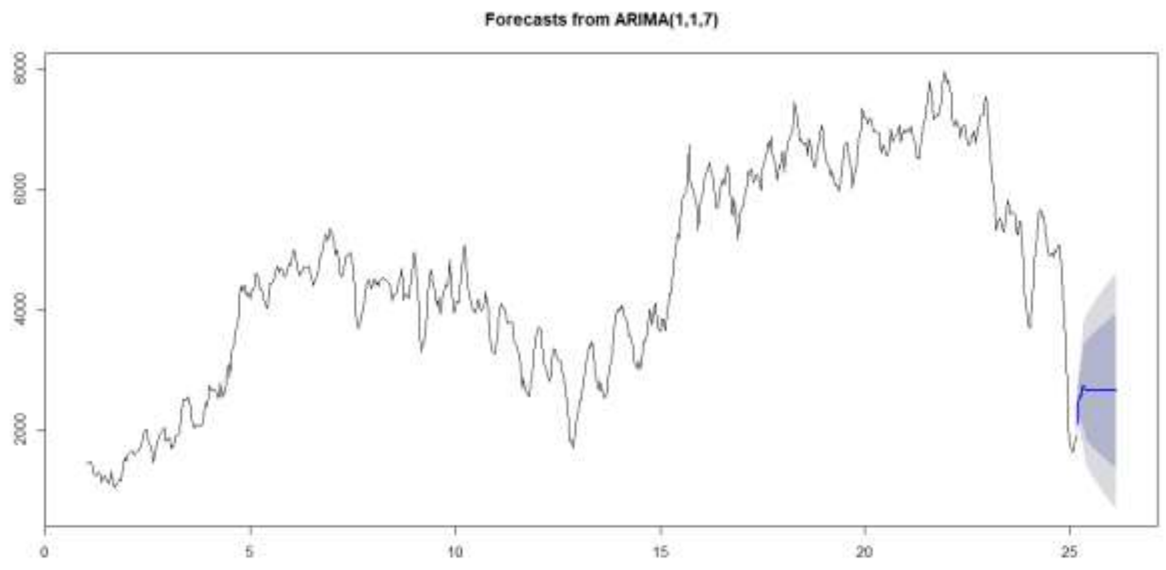


Figure: Forecast

The light blue line above shows the fit provided by the model, but what if we wanted to get a sense of how the model will perform in the future? One method is to reserve a portion of our data as a "hold-out" set, fit the model, and then compare the forecast to the actual observed values:

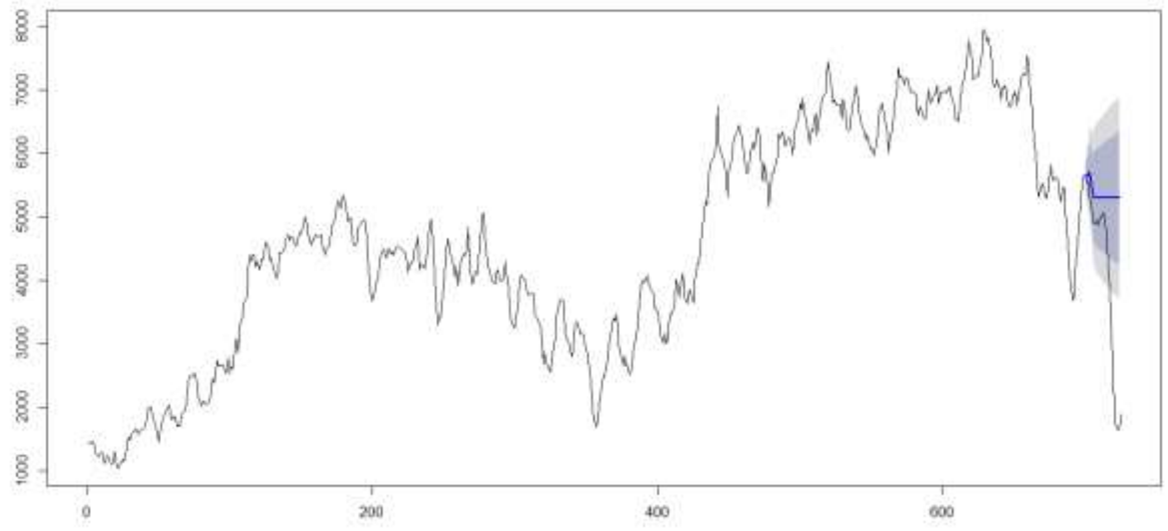


Figure: Forecast

However, the blue line representing forecast seems very naive: It goes close to a straight line fairly soon, which seems unlikely given past behaviour of the series. Recall that the model is assuming a series with no seasonality, and is differencing the original non-stationary data. In other words, plotted predictions are based on the assumption that there will be no other seasonal fluctuations in the data and the change in number of bicycles from one day to another is more or less constant in mean and variance. This forecast may be a naive model, but it illustrates the process of choosing an ARIMA model and could also serve as a benchmark to grade against as more complex models are built.

How can we improve the forecast and iterate on this model? One simple change is to add back the seasonal component we extracted earlier. Another approach would be to allow for (P, D, Q) components to be included to the model, which is a default in the auto.arima() function. Re-fitting the model on the same data, we see that there still might be some seasonal pattern in the series, with the seasonal component described by AR(1):

Note that (p,d,q) parameters also changed after we included a seasonal component. We can go through the same process of evaluating model residuals and ACF/PACF plots and adjusting the structure if necessary. For example, we notice the same evidence of higher order is present in the auto correlations with lag 7, which suggests that a higher-order component might be needed:

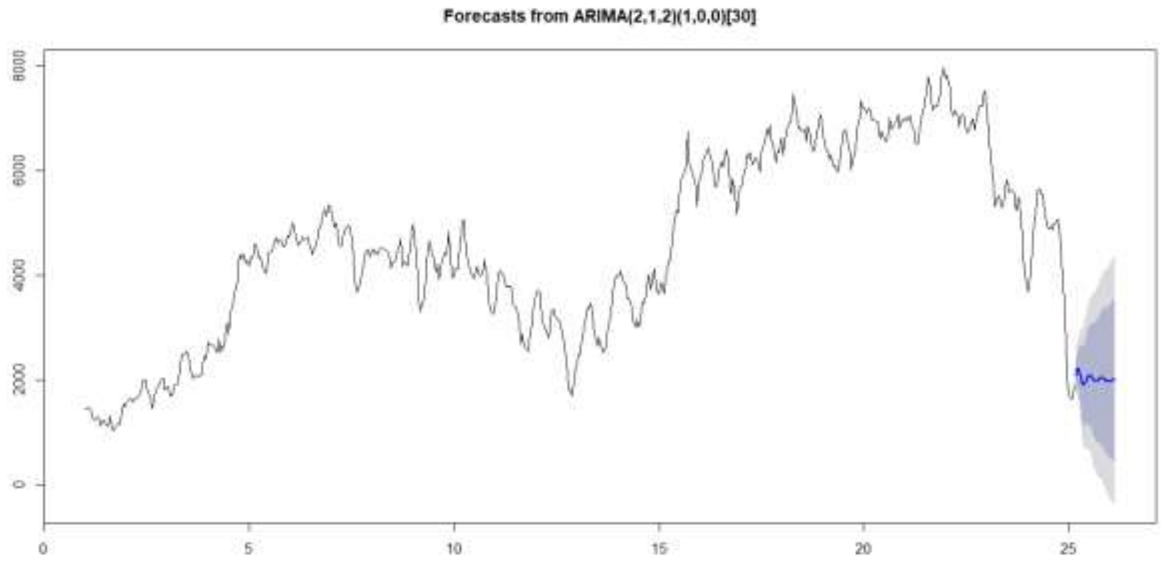


Figure: Forecast considering seasonality

Both forecast estimates above are provided with confidence bounds: 80% confidence limits shaded in darker blue, and 95% in lighter blue. Longer term forecasts will usually have more uncertainty, as the model will regress future Y on previously predicted values. This is reflected in the shape of the confidence bounds in this case, as they start to widen with increasing horizon. The pattern in confidence bounds may signal the need for a more stable model. It is very important to look at prediction bounds and keep in mind the expected error associated with point estimates.

Chapter 3

Conclusion

After an initial naive model is built, it's natural to wonder how to improve on it. Other forecasting techniques, such as exponential smoothing, would help make the model more accurate using a weighted combinations of seasonality, trend, and historical values to make predictions. In addition, daily bicycle demand is probably highly dependent on other factors, such weather, holidays, time of the day, etc. One could try fitting time series models that allow for inclusion of other predictors using methods such ARMAX or dynamic regression. These more complex models allow for control of other factors in predicting the time series.

References

Arima and Cointegration Tests of PPP under Fixed and Flexible Exchange Rate Regimes

ARIMA model for forecasting oil palm price

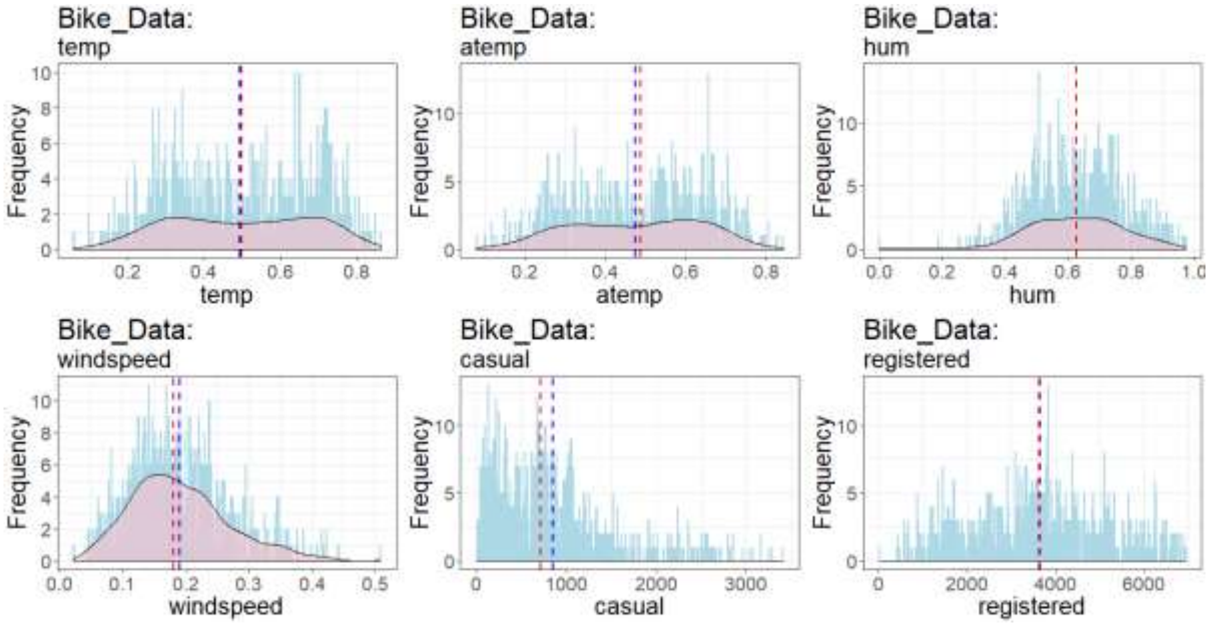
https://towardsdatascience.com/an-end-to-end-project-on-time-series-analysis-and-forecasting-with-python-

https://medium.com/@josemarcialportilla/using-python-and-auto-arima-to-forecast-seasonal-time-series-90877adff03c

https://www.digitalocean.com/community/tutorials/a-guide-to-time-series-forecasting-with-arima-in-python-3

Extra Figures

Histograms of the predictor variables with red dotted line indicating median and blue dotted line indicating mean


Bike_Data:
temp
Frequency
temp
Bike_Data:
atemp
Frequency
atemp
Bike_Data:
hum
Frequency
hum
Bike_Data:
windspeed
Frequency
windspeed
Bike_Data:
casual
Frequency
casual
Bike_Data:
registered
Frequency
registered