DS 901 - Project Elective

Predicting Bunching of BMTC Buses

Guide: Prof. G.Srinivasaraghavan

Members:
Bhagirathi Hegde (MT2016034)
Jyotsana (MT2016068)
M Chellapriyadharshini (MT2016041)

Contents:

Problem Description:

"Bunching" is a scenario when two or more buses along the same bus route arrive together or a short distance from each other instead of being evenly spaced apart. This may happen due to several reasons-when one or more of the buses are unable to adhere to their schedule or when there is an unexpected break down of a vehicle or due to the speed policies of individual drivers. This results in longer waiting times for the passengers and an overall resentment about the bus service.

Proposed Solution Description:

In this project we intend to predict those bunching points for the buses of Bangalore Metropolitan Transport Corporation (BMTC) so as to prevent it from happening by scheduling delays in real time. We are provided with the data transmitted by the GPS devices fitted in the BMTC buses. Using this we try to arrive at a model that best depicts the characteristics of the buses that "bunch".

Data Description:

The National Marine Electronics Association (NMEA) has developed a specification that defines the interface between various pieces of marine electronic equipment. The standard permits marine electronics to send information to computers and to other marine equipment.

GPS receiver communication is defined within this specification. The data emitted by a GPS device is in the form of NMEA sentences. There are different NMEA sentence types. One of which is the GPRMC sentence-where 'GP' denotes that it contains GPS data and 'RMC' denotes Recommended Minimum sentence C, which is NMEA's own version of essential GPS position, velocity and time data. Every GPRMC sentence contains Latitude, Longitude, Speed, Track Angle, Date, Magnetic Variation, Checksum, etc.

The data transmitted by the GPS device fitted in a BMTC bus, is also received in this form. This data is then parsed to extract meaningful information and stored in a single RDBMS table VTS_PARSE_DATA. This table contains the following columns:

Table_ID	Country_Code	IST_Time	Longitude_Degree	Variation_Sense
ID	Network_Code	Digital_Output	EW	Magnetic_Variation
Packet_Header	Location_Area_Code	Analog_Input	Longitude	Checksum_Hex
Device_ID	Cell_Id	Max_Speed	Vehicle_Direction	Parser_Version
Packet_Code	No_Satellite_In_View	GPRMC_Header	Speed_Knots	Server_Port
Misc_Bytes	Ext_Battery_Voltage	Data_Status	Speed_KMPH	Parser_Version_Date
IGN_Status	Internal_Battery_Vtg	Lat_Degree	GMT_Time	Created_Date
Acc_Distance	Sensor_Info	NS	GMT_Date	Record_Status
Signal_Strength	Digital_Input	Lat	IST_Date	

Table: VTS_PARSE_DATA Schema

The VTS_PARSE_DATA table contains data for over 3 months. It is this table's data that we are provided with. The field that uniquely identifies every bus in this data is the "Device_ID" attribute, which is a unique ID assigned to every GPS device.
In addition we are also provided with two mapping files:

1. Vehicle_Device_Relation.xls: This file gives the relationship between the vehicle and the GPS device. That is which GPS Device – identified by its Device_ID is fitted in which Bus-identified by its License Plate number. It also gives the SIM number associated with every Device_ID.
2. Schedule_Mapping_Vehicle.xls: This file gives the Depot_Name and Schedule_Name associated with every bus (once again identified by the license plate number).
   The Schedule_Name is of the format "Route_Number/Bus_Number-Days Plying".

Visual and Exploratory Analysis of the data:

Equipped with the data available in the VTS_PARSE_DATA table, Vehicle_Device_Relation and Schedule_Mapping_Vehicle files, we proceeded to analyze it in the following manner:

- To start with, we decided on a route along which we were to model and hence predict bunching. The route was decided to be from HSR BDA Complex to Electronic City Wipro Gate and among the buses that ply on this route, we decided to work on the 356CW buses' data.
- Having decided on this, we consulted the Schedule_Mapping_Vehicle file to find all the records whose Schedule_Name start with "356CW". There were 19 such buses. We thereby found the License Number and Depot_Name associated with every 356CW bus.
- Then we filtered out the records in the Vehicle_Device_Relation file based on the License Numbers obtained in the previous step. And therefore obtained the GPS Device_IDs associated with every 356CW bus.
- Having obtained the required Device_IDs, we then queried the database table VTS_PARSE_DATA in order to get the records filtered by the Device_ID. Among the 44 attributes, we queried for the following:
  - Device_ID
  - Lat
  - Longitude
  - GMT_Date
  - GMT_Time
- We repeated this exercise for the 19 buses that had been identified as 356CW and obtained the data in csv files, one for each Device_ID.
- So we then had the Lat-Long data for the 19 buses whose Schedule_Name begins with 356CW. But it was here we realized that all buses that were marked 356CW do not follow the same route. And that the route depended upon the Depot to which a specific bus belonged to. Also the "route" itself-specifying the origin and destination-is not available from any of the files or from the database table. That is though the Schedule_Name in the Schedule_Mapping_Vehicle file gives the route number as 356CW and the associated Depot_Name, the actual origin and destination points, like "HSR BDA Complex" to "Electronic City Wipro Gate", is not mentioned anywhere.
- Hence we grouped the 19 buses according to their corresponding Depot_Name.
- Also, the Depot_Names are of the form "Depot-XX" where XX is an integer. They do not give the names of the places where the depot is located. So it was not possible for us to identify which bus operated on the required route.

- Therefore, among the 19 buses of 356CW, in order to identify those buses that ply on the route "HSR BDA Complex to Electronic City Wipro Gate", we had to plot a few latitude-longitude points of each of them on Google Maps.
- By carrying out this exercise, we identified 4 Device_IDs (i.e. 4 buses) that operate on the required route. The details of those are as below:

Depot_Name	**Device_ID**	**License Plate Number**
Depot-19	150219795	KA57F0477
Depot-19	150220249	KA01F9413
Depot-38	150813511	KA57F0945
Depot-38	150814233	KA57F0944

Table: 356CW Bus Details

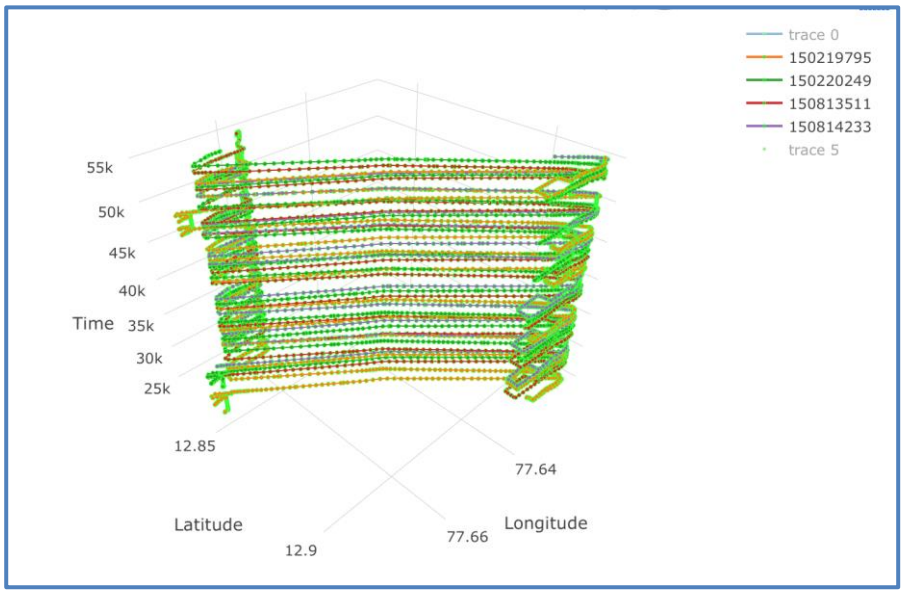
- So now our dataset was reduced to contain the Latitude, Longitude, GMT_Date and GMT_Time for these 4 Device_IDs.

Once we had the required data, we made the following observations:

- For a given day, we have the Latitude Longitude positions of a bus, approximately, for every 10 seconds.
- GPS transmissions are available from 12:00 AM.
- During the time period from 12:00 AM to around 5:30 AM the interval between successive data points is erratic. After that we get the data points for every 10 seconds.
- Another observation was that just by looking at the data it was not possible for us to segregate them into trips – that is when the bus completes one round trip and starts the next one.

Based on our understanding of the data, we decided to do the following as initial steps:

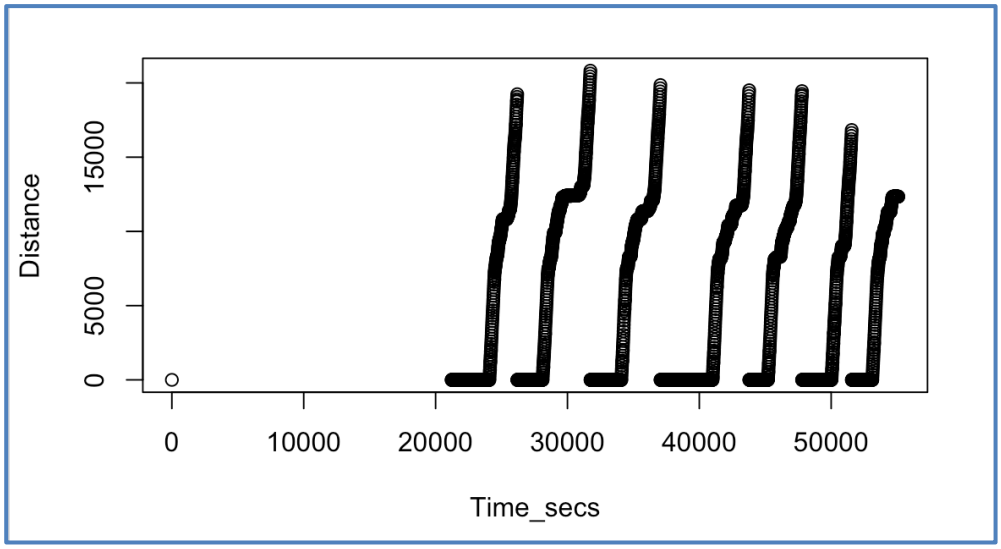
- Time: The GMT_Time was in the "HH:MM:SS" format. We converted it into seconds by applying the formula:
  - TIME_IN_SECS = HH*3600 + MM*60 + SS
- Origin Point: Since we wanted to divide the data points into multiple trips, we decided to fix a Latitude-Longitude position as our "Origin Point" from which our round trip would be calculated. We selected the Lat-Long corresponding to the stop "Electronic City Wipro Gate" as our origin point. The position of which is given by:
  - Origin Point – Latitude: 12.837455
  - Origin Point – Longitude: 77.658393
- Lat-Long-Time Plot: We also created a 3-dimensional plot using the Latitude, Longitude and the Time (in seconds) in order to get a visual understanding of the data.



Bunching Algorithm and Implementation:

The steps we followed in identifying the bunching points are:

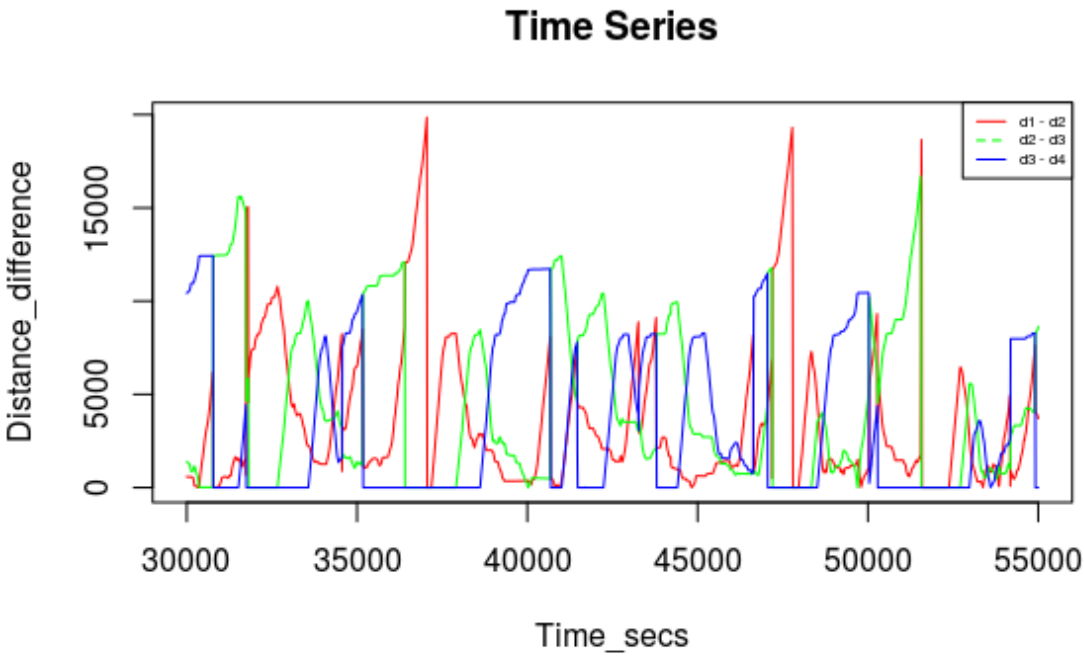
1) Divide the data points into multiple round trips from the origin point. This is done by matching current latitude-longitude with the origin point, if it matches then we say that it is a new trip i.e., it has completed a round trip.
2) Calculate the distance covered incrementally, by computing the Haversine distance between the adjacent latitude-longitude points and adding it to the distance covered from the origin so far. Each time a round trip is completed reset the distance.
    a) Haversine distance is the shortest distance between 2 points on the spherical surface of earth, determined by its latitude and longitude, measured along its surface.
3) Compute the 2-dimensional plots of Distance vs. Time for each trip. For one Device_ID the 2-D plot looks as below:



4) Combine these computed values into a single table - call it 'dist_time_table' - wherein-for every 10th second, for each of the 4 Device_IDs, interpolate (in case

the data is not available for every 10 seconds) the distance covered by it from the origin.

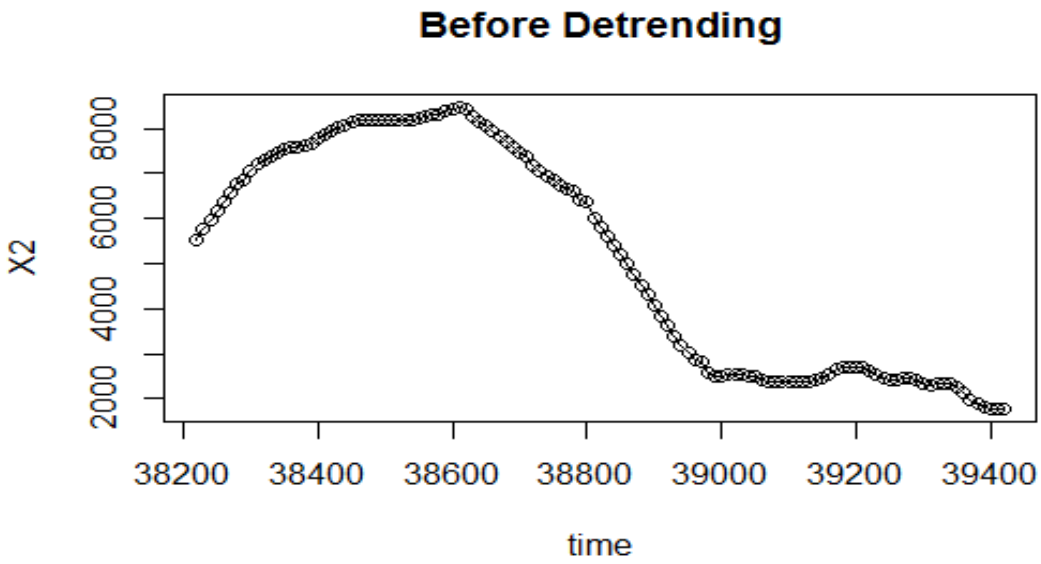
5) Determine the bunching points by pairwise comparing the difference between the distances of the buses. If the difference between the 2 buses is less than or equal to 20 meters then bunching is assumed to happen.
6) Now in the dist_time_table, for each row, sort the distances in descending order. Then find the differences between the sorted distances. By doing this, at every time instant we are getting the distance between the bus that is farthest from the origin and the bus that trails behind that; the distance between the bus that is second farthest from the origin and the bus that is trailing behind that; and so on.
7) In our case, since we have 4 buses, we get 3 such columns of distance gaps.
8) Plot each of these columns as a Time Series plot.



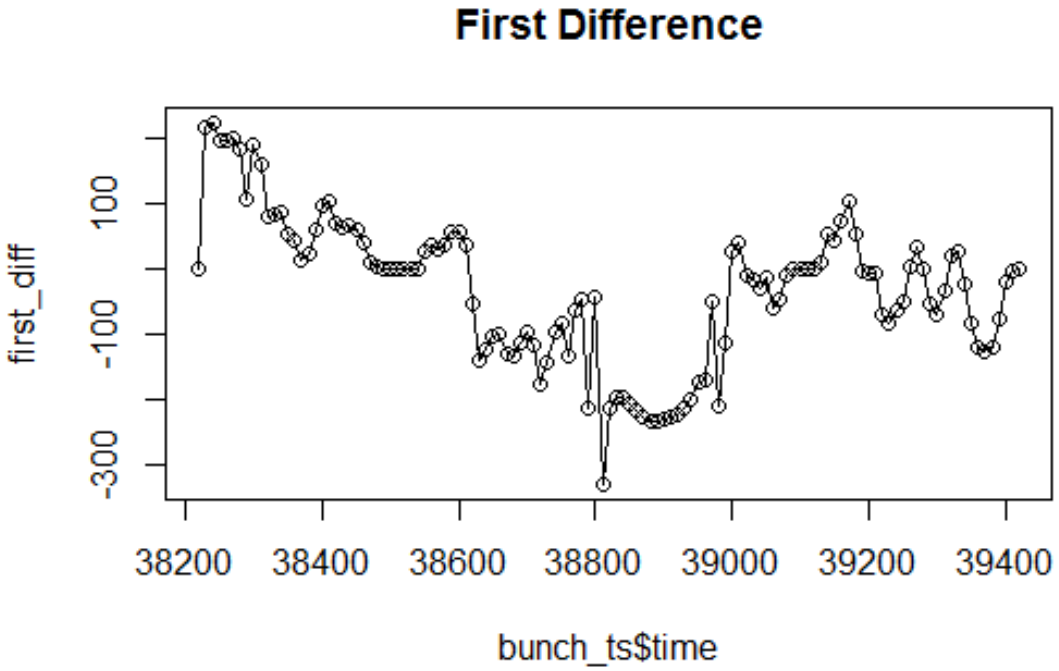
9) The points where distance gaps become zero are bunching points. Note that these points may include the case when both buses are at origin together, in that case they are ignored.
10) Now segment the data such that it contains 30 minutes of data before bunching occurred. This constitutes the training set for time series analysis.

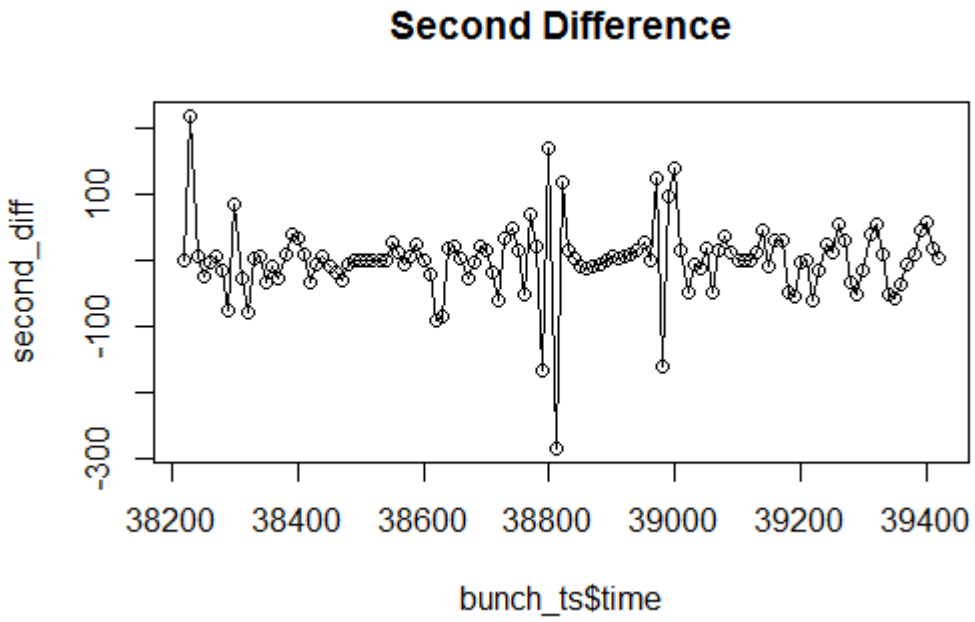
Time Series Modelling and Analysis:

1) Assumption: No seasonality and cyclicity in the data as the time duration is very short.

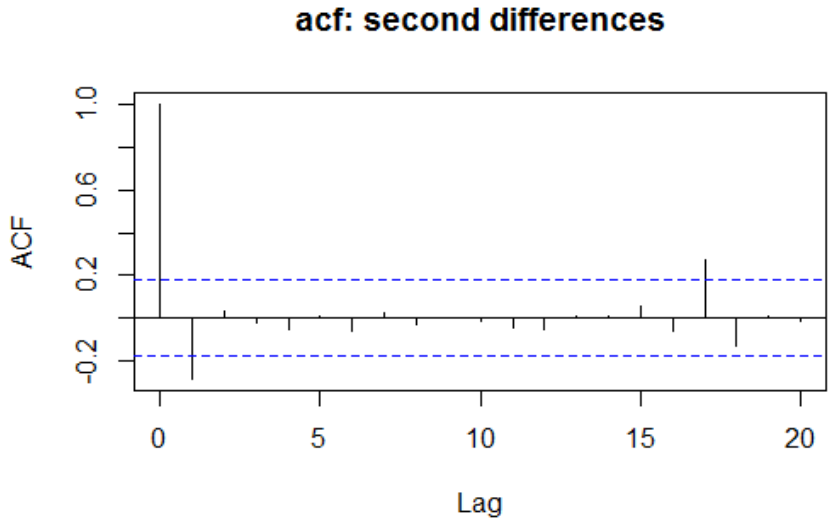


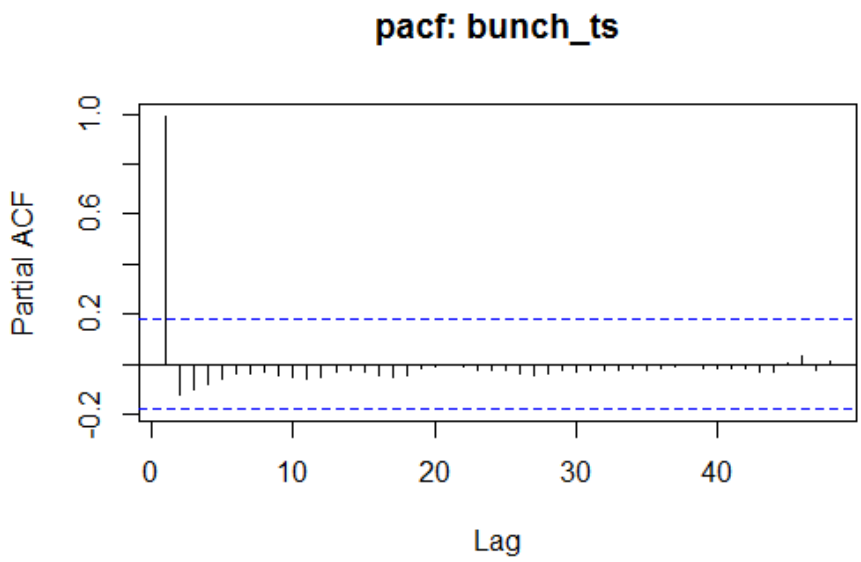
2) Make the data stationary, required for modelling the time series, by removing the trend in the series. For our data, as seen in the figures below, taking difference of difference of data points detrends the data.



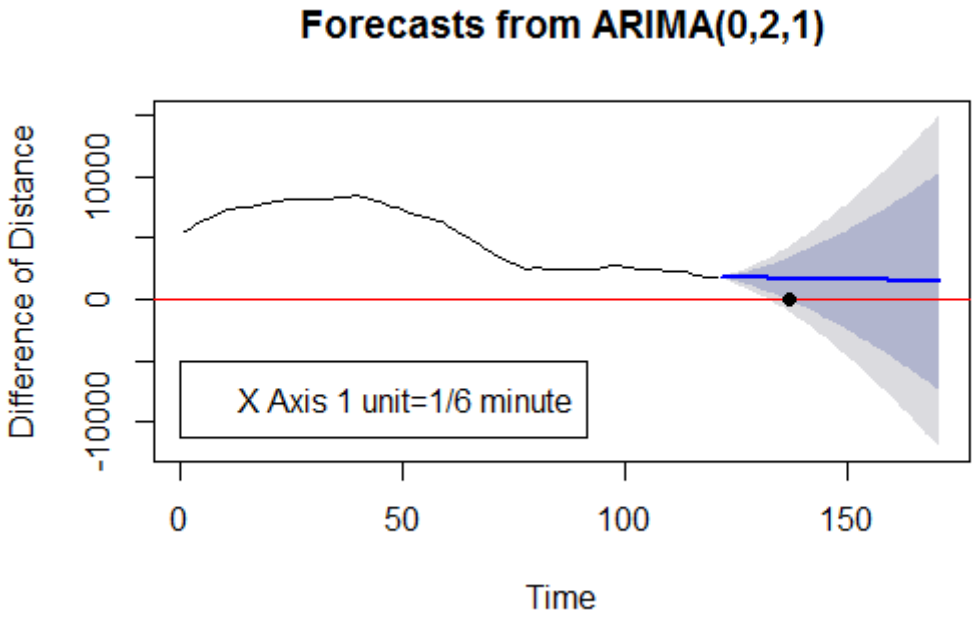


3) Plot the PACFs and ACFs for the original time series, its first and second difference to see which arima model (p, d, q) fits the best; where p, d and q are the orders of AR (Auto Regression), differences and MA (Moving Average).
The following figures show PACF and ACF plots for one of the times series data.





4) Fit an arima model to the detrended data, which predicts the bunching.



The above forecast tells us with 95% confidence that the bunching will occur at 137 unit time. This can be seen by drawing a horizontal line from the origin and taking the first intersection with the 95% confidence interval. (The inner dark blue region corresponds to the 95% confidence interval in which the predicted value might lie.)

Code:

Statistical Programming Language-R was used to do the exploratory and time series analysis on the data. The R scripts can be found in the github repository.

Results:

We have predicted bunching for 3 pairs of buses shown below.

1) **Predicted bunching** point after 2.83 minutes with 95% confidence and 4.5 minutes with 99% confidence.
   Actual bunching point exactly after 10 minutes.

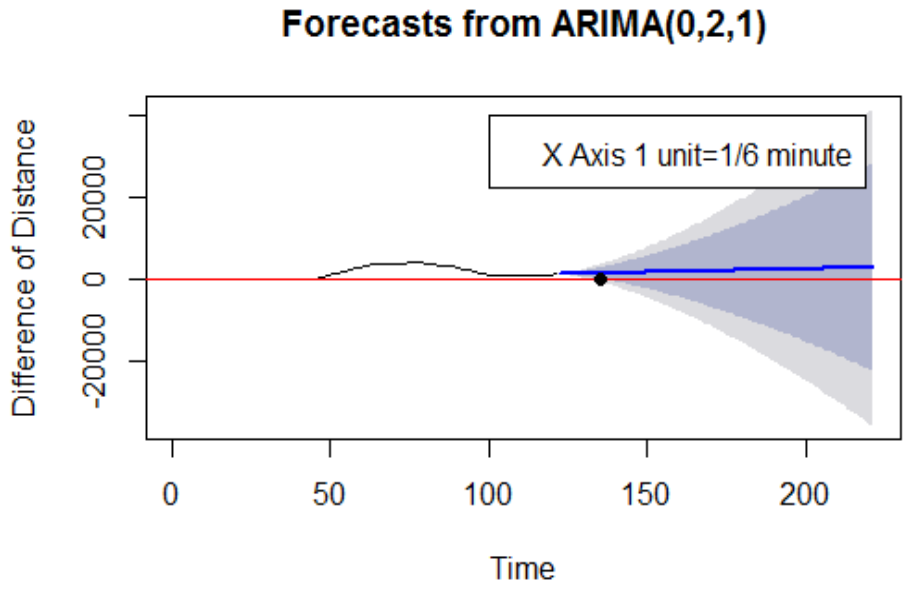


Figure: Bunching Point at 95% confidence

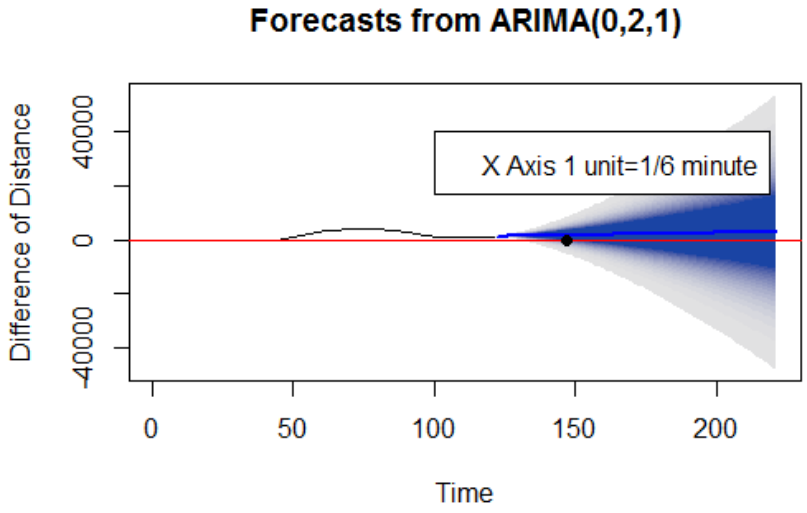


Figure: Bunching Point at 99% confidence

2) **Predicted bunching** point after 2.83 minutes with 95% confidence and after 4.2 minutes with 99% confidence.
Actual bunching point exactly after 10 minutes.

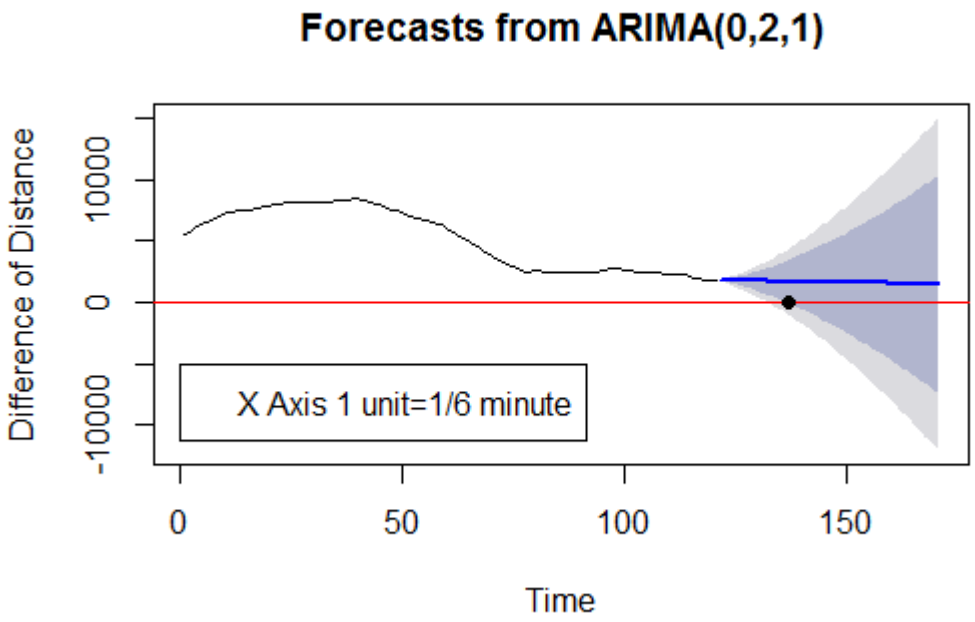


Figure: Bunching Point at 95% confidence

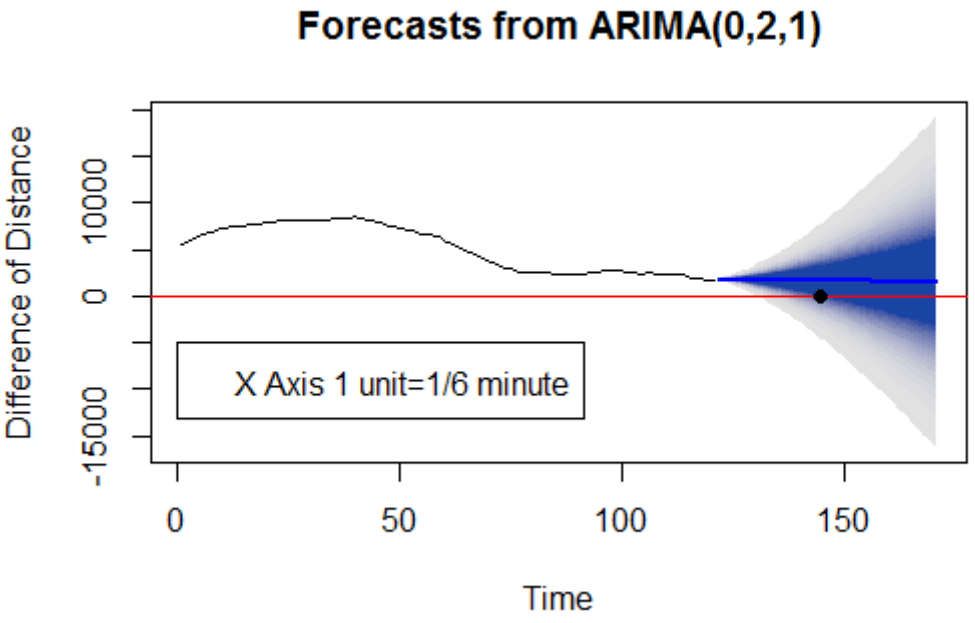


Figure: Bunching Point at 99% confidence

3) **Predicted bunching** point after 2.5 minutes with 95% confidence and after 3.84 minutes with 99% confidence.
Actual bunching point exactly after 10 minutes.

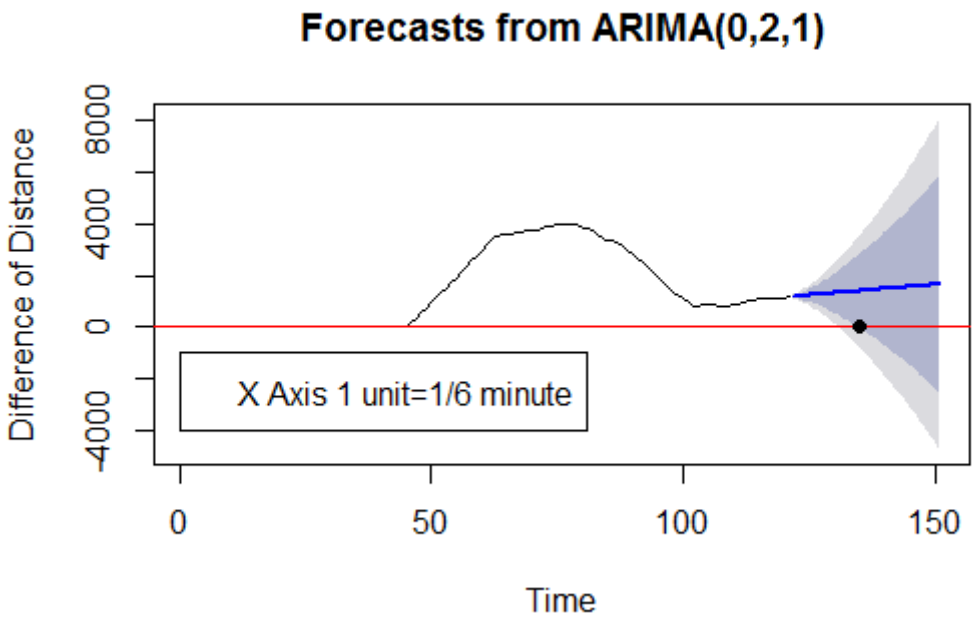


Figure: Bunching Point at 95% confidence

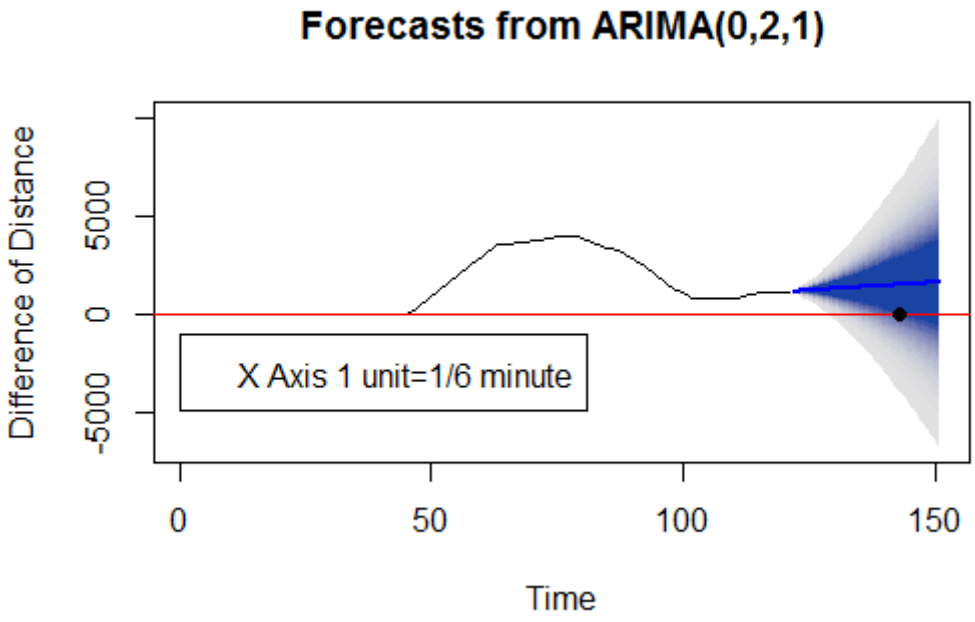


Figure: Bunching Point at 99% confidence