Assignment 4: ARIMA and sARIMA

Peter Ye

2024-04-15

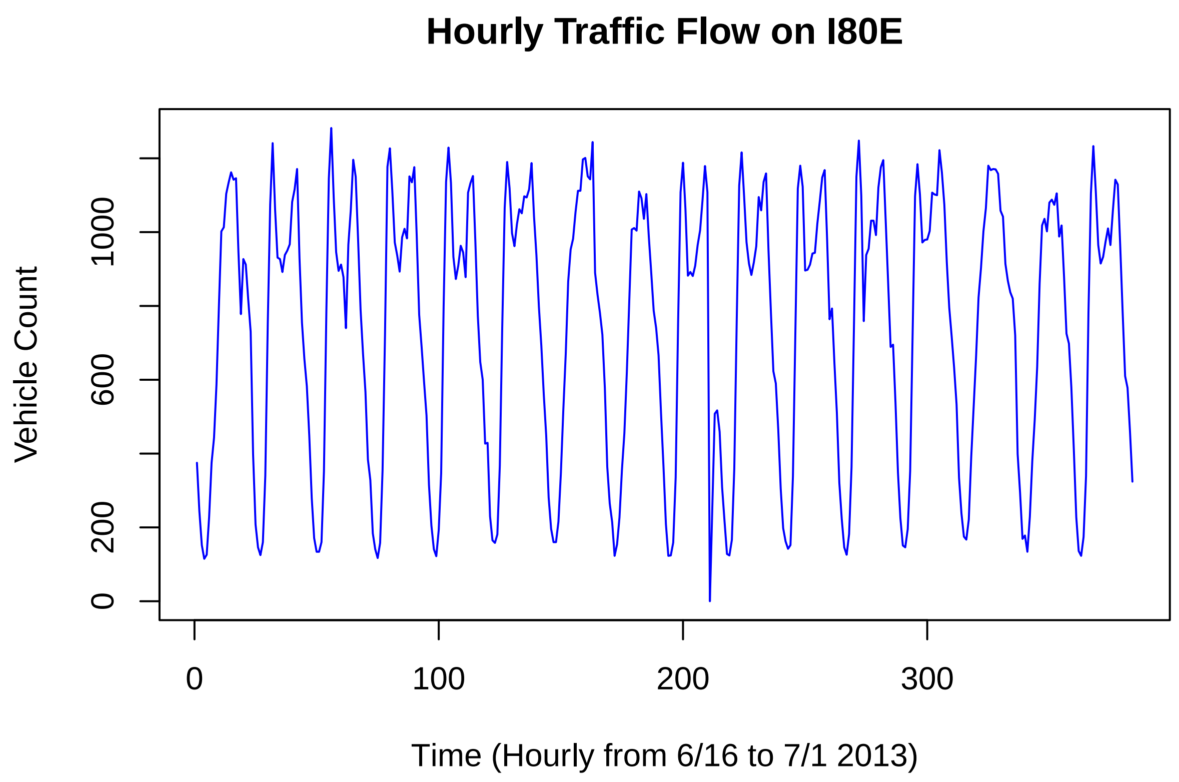
Question 1

```
# Use Python to combine file automatically

import os
import pandas as pd
from datetime import datetime

# Set the directory
directory = "Traffic Flow Data"

# Function to parse date from filename
def parse_date(filename):
    parts = filename.split('-')
    year = int(parts[2])
    month = parts[3]
    day = int(parts[4].split('.')[0])  # Remove the file extension
    month_number = datetime.strptime(month, '%B').month
    return datetime(year, month_number, day)

# List to hold data from each file
data_frames = []

# Get all Excel files and sort them by date in descending order
files = [f for f in os.listdir(directory) if f.endswith(".xls") or f.endswith(".xlsx")]
files.sort(key=parse_date, reverse=False)  # Sort ascending to maintain the file order by date

# Iterate through every file in the sorted list
for filename in files:
    file_path = os.path.join(directory, filename)
    # Determine the appropriate engine based on file extension
    engine = 'xlrd' if filename.endswith('.xls') else 'openpyxl'
    try:
        data = pd.read_excel(file_path, sheet_name='Sheet0',
        usecols="E", skiprows=4, nrows=24, engine=engine)
        data_frames.append(data)
    except Exception as e:
        print(f"Failed to process {filename}: {e}")

# Only attempt to concatenate if data_frames is not empty
if data_frames:
    combined_data = pd.concat(data_frames, axis=0)  # Concatenate vertically
    combined_data.columns = ['traffic_flow']  # Rename the single column to 'traffic_flow'
    combined_data.reset_index(drop=True, inplace=True)  # Reset index to ensure continuous index
    output_path = os.path.join(directory, 'Combined_Traffic_Flow_Data.xlsx')
    combined_data.to_excel(output_path, index=False, engine='openpyxl')
    print(f"Data combined and saved to {output_path}")
else:
    print("No valid Excel data to process.")
```

```
## Data combined and saved to Traffic Flow Data\Combined_Traffic_Flow_Data.xlsx
```

```
# Read the combined excel
traffic <- read_xlsx("Traffic Flow Data/Combined_Traffic_Flow_Data.xlsx")
```

```
# Convert it to TS object
traffic_ts <- ts(traffic$traffic_flow, start=c(1), end=c(384), frequency = 1)
plot(traffic_ts, xlab = "Time (Hourly from 6/16 to 7/1 2013)", ylab = "Vehicle Count",
     main = "Hourly Traffic Flow on I80E", col = "blue", type = "l")
```



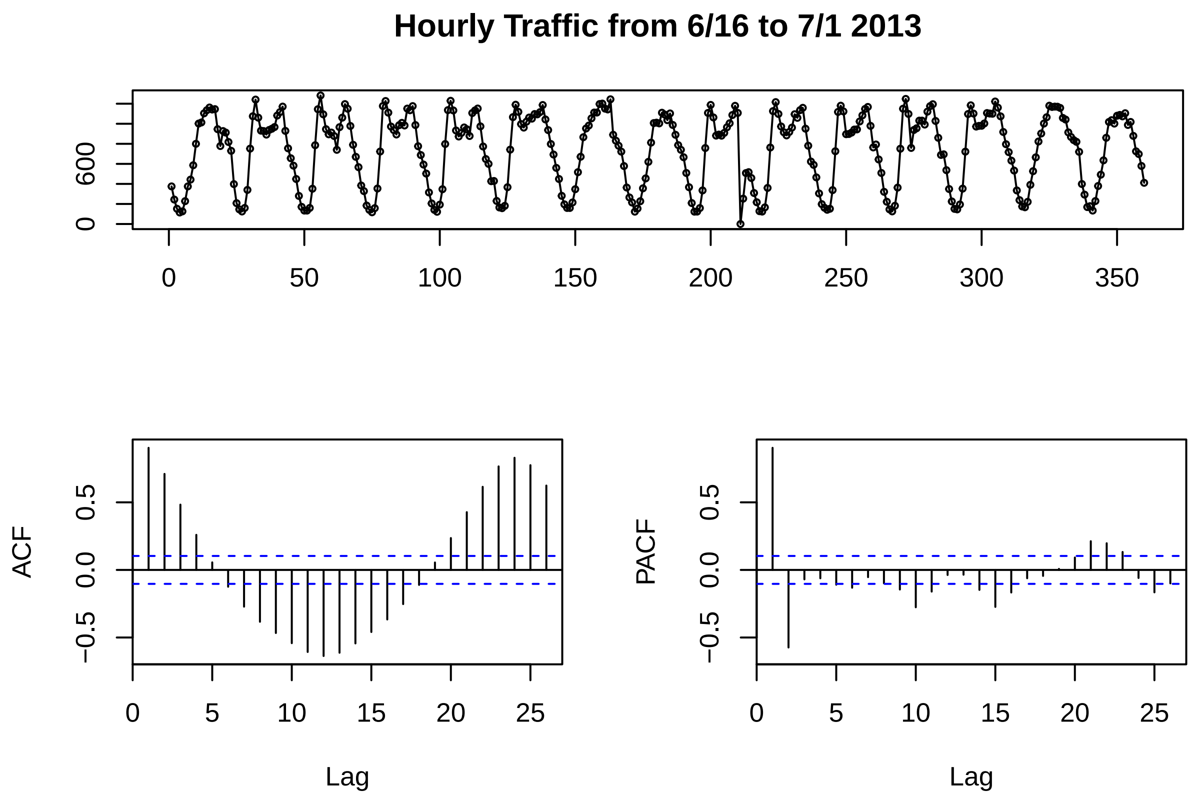
- Trend: : The plot does not appear to show a long-term upward or downward trend, which would indicate an overall increase or decrease in traffic volume over the period observed. The data seems to fluctuate around a constant mean, suggesting stable average traffic counts from June 16, 2013, to July 1, 2013.

- Seasonality: The plot displays a very consistent pattern that repeats approximately every 24 hours. This is indicative of daily seasonality, which is a common characteristic in traffic flow data. The regularity of the pattern suggests that daily traffic behaviors are consistent across the observed period.
- Variability: There is some variability in the height of the peaks and depths of the troughs, which could be due to variability in daily traffic patterns or could correspond to specific days of the week.

Question 2

```r
# split the data
train <- traffic[1:360,]
test <- traffic[361:384,]

# Covert the train data into time series object
train_ts <- ts(train$traffic_flow, start = c(1), end=c(360), frequency = 1)
```

```r
tsdisplay(train_ts,main='Hourly Traffic from 6/16 to 7/1 2013')
```



- ACF: The slow decay of the ACF also suggests a potential non-stationarity in the mean of the time series, which could indicate that differentiating the series might be necessary if building a model.
- PACF: It exhibits significant spikes at the first few lags, and then it mostly falls within the confidence interval. This suggests that there is some dependency between an hour and its immediate past hours that is not explained by the overall daily seasonality.

Question 3

Check BoxCox

```
## [1] 1.084656
```

The lambda is 1.084656 and close to 1, suggesting BoxCox transformation is not necessary

Check KPSS

```
kpss <- kpss.test(train_ts)
kpss
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  train_ts
## KPSS Level = 0.025757, Truncation lag parameter = 5, p-value = 0.1
```

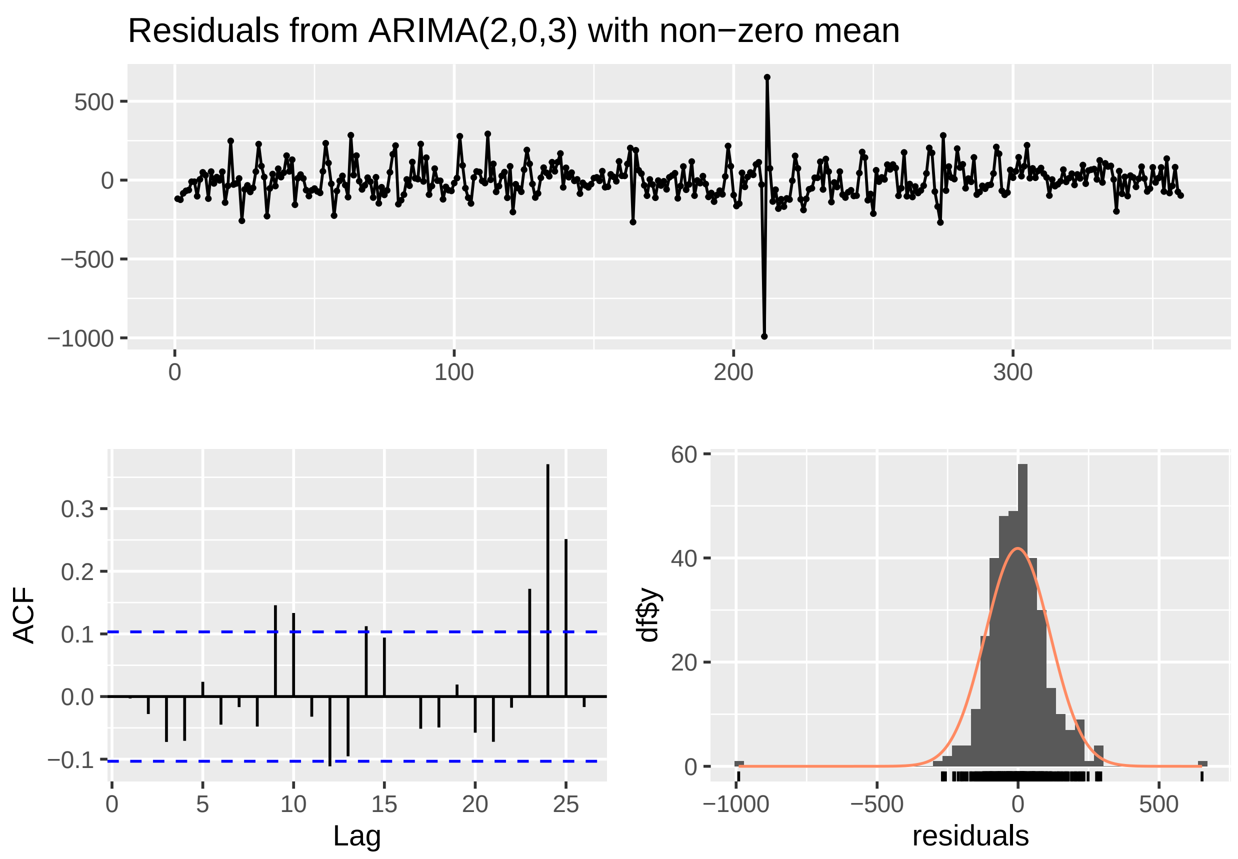
The KPSS level at 0.025757 with p-value of 0.1, indicating the data is stationary

auto-ARIMA model with AICc

```
# Fit the auto-ARIMA model with AICc
auto_arima_model_aicc <- auto.arima(train_ts, ic ="aicc")
summary(auto_arima_model_aicc)
```

```
## Series: train_ts
## ARIMA(2,0,3) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1      ma2      ma3      mean
##       1.8088  -0.8853  -0.5348  -0.2671  -0.1157  746.3181
## s.e.  0.0288   0.0287   0.0600   0.0596   0.0654    6.8586
##
## sigma^2 = 13443:  log likelihood = -2220.78
## AIC=4455.56   AICc=4455.88   BIC=4482.77
##
## Training set error measures:
##                     ME     RMSE    MAE  MPE MAPE      MASE         ACF1
## Training set -1.390098 114.9732 79.019 -Inf  Inf 0.7027304 -0.003018285
```

```
# Residuals check
checkresiduals(auto_arima_model_aicc)
```



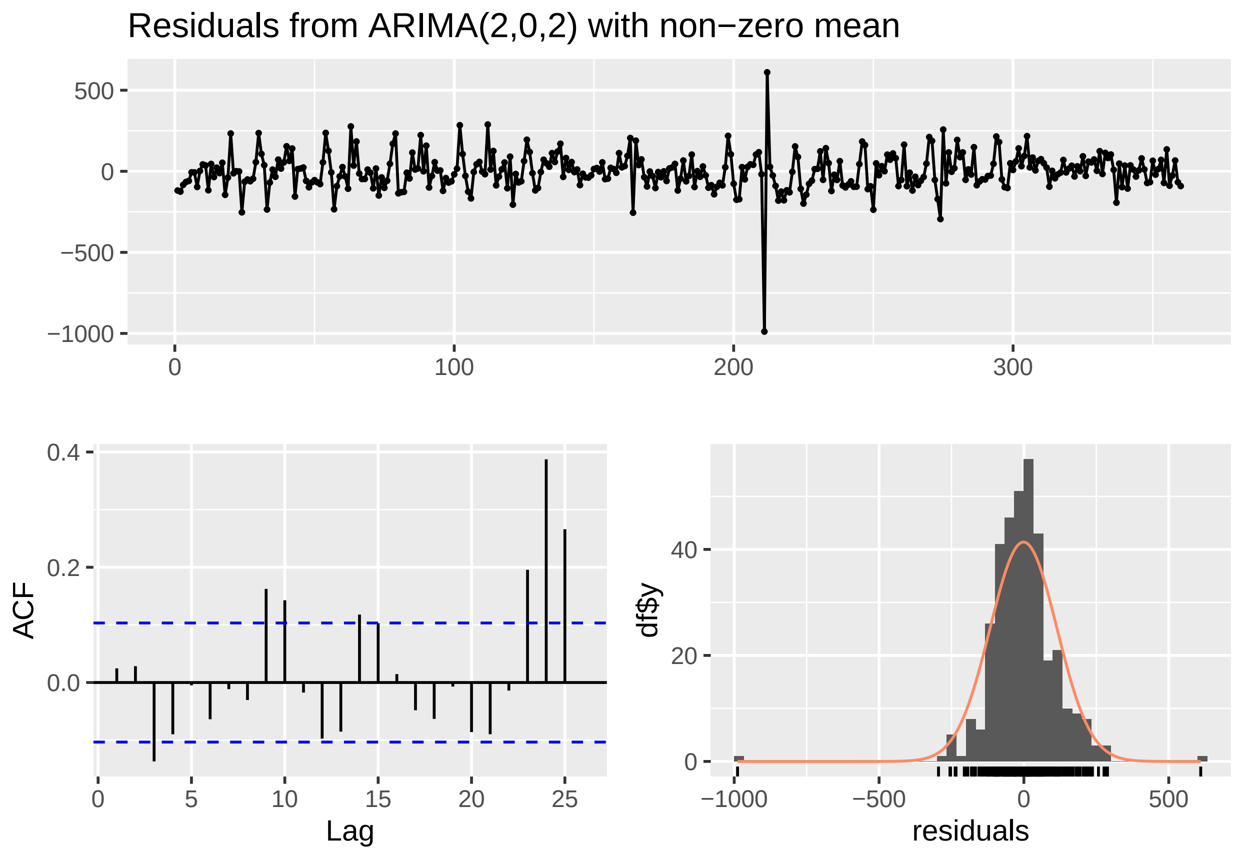
```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,3) with non-zero mean
## Q* = 20.439, df = 5, p-value = 0.001033
##
## Model df: 5.   Total lags used: 10
```

auto-ARIMA model with BIC

```
# Fit the auto-ARIMA model with BIC
auto_arima_model_bic <- auto.arima(train_ts, ic ="bic")
summary(auto_arima_model_bic)
```

```
## Series: train_ts
## ARIMA(2,0,2) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1      ma2      mean
##       1.8308  -0.9072  -0.5916  -0.3254  746.3649
## s.e.  0.0229   0.0228   0.0488   0.0471    6.9120
##
## sigma^2 = 13514:  log likelihood = -2222.26
## AIC=4456.52   AICc=4456.76   BIC=4479.83
##
## Training set error measures:
##                     ME     RMSE      MAE  MPE MAPE      MASE       ACF1
## Training set -1.376334 115.4405 79.43987 -Inf  Inf 0.7064733 0.02457699
```

```
# Residuals check
checkresiduals(auto_arima_model_bic)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,2) with non-zero mean
## Q* = 29.589, df = 6, p-value = 4.704e-05
##
## Model df: 4.   Total lags used: 10
```

AICc suggests the ARIMA(2,0,3) model while BIC suggests the ARIMA(2,0,2) model. The Ljung-Box test shows a p-value of 0.001033,indicating that there are still autocorrelations in the residuals at lag 10 that the model has not captured. Similarly, the Ljung-Box test for this model also shows a very low p-value 4.704e-05, indicating significant autocorrelation in the residuals.

EACF

```
eacf(train_ts)
```

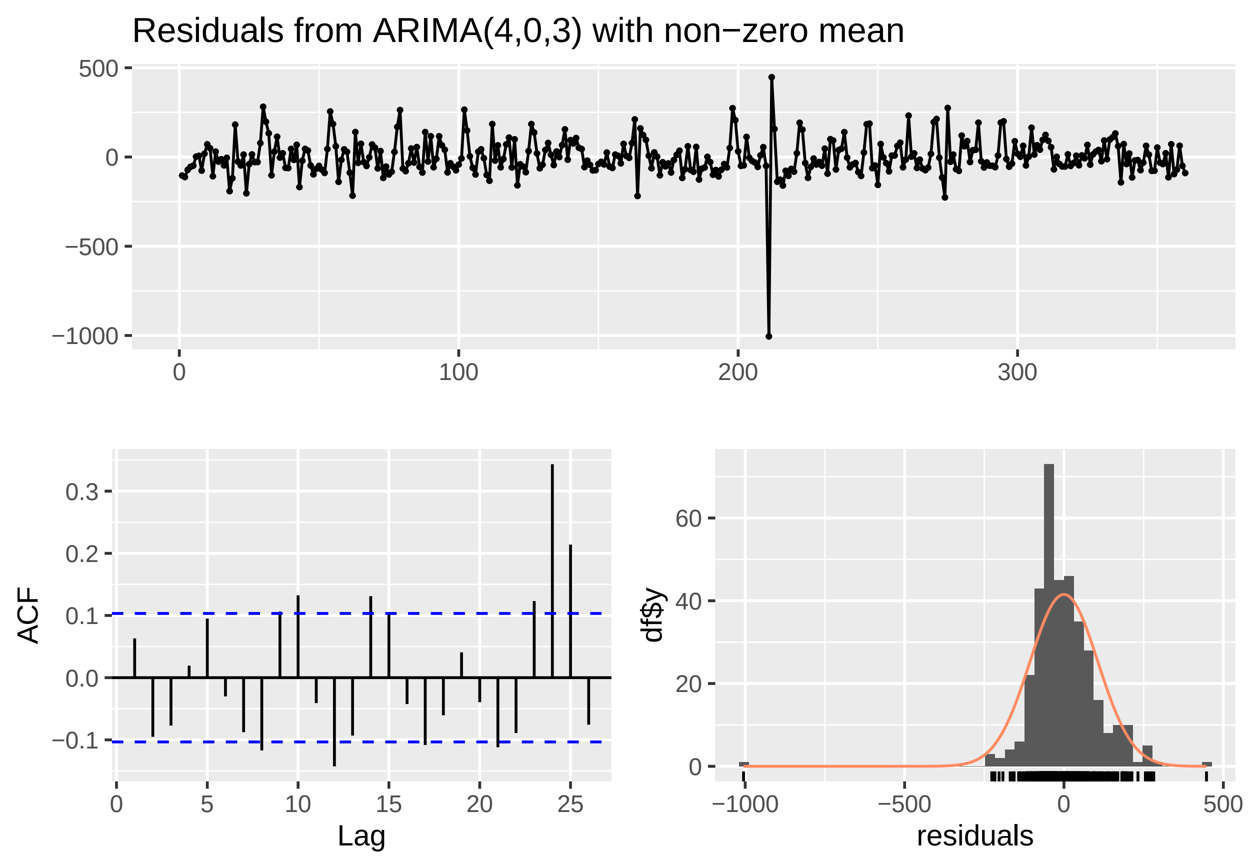
```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x o x x x x x x  x  x  x
## 1 x x x x o x x x o o x  x  x  x
## 2 o o x o o o o x x o o  x  x  x
## 3 x o o x o o o o o o o  o  x  o
## 4 x x o x o o o o o o o  o  o  o
## 5 x x x o o o o o o o o  o  o  o
## 6 x x x x o o o o o o o  o  o  o
## 7 x x o x x o o o o x o  o  o  o
```

The dot on the EACF suggests the ARIMA(4,0,3) is a possible solution.

```
# Create Arima(4,0,3) model
model_manual = Arima(train, order=c(4, 0, 3))
# Check the autocorrelation
summary(model_manual)
```

```
## Series: train
## ARIMA(4,0,3) with non-zero mean
##
## Coefficients:
##          ar1      ar2     ar3      ar4      ma1     ma2      ma3      mean
##       3.4090  -4.6365  2.9893  -0.7825  -2.3609  1.8743  -0.4778  743.2782
## s.e.  0.1766   0.4835  0.4519   0.1428   0.3014  0.6100   0.3282    9.8632
##
## sigma^2 = 11649:  log likelihood = -2195.46
## AIC=4408.92   AICc=4409.44   BIC=4443.9
##
## Training set error measures:
##                      ME     RMSE      MAE  MPE MAPE      MASE       ACF1
## Training set -0.1743657 106.7253 73.46767 -Inf  Inf 0.6533614 0.06318195
```

```
# Check residuals
checkresiduals(model_manual)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(4,0,3) with non-zero mean
## Q* = 29.336, df = 3, p-value = 1.904e-06
##
## Model df: 7.   Total lags used: 10
```

The ARIMA(4,0,3) models fail to fully capture the underlying patterns in the data, as evidenced by significant autocorrelation and non-normality in the residuals.

Question 4

```r
# Create the time series object
train_ts_weekly <- ts(train$traffic_flow, frequency = 168)
auto_arima_model_weekly <- auto.arima(train_ts_weekly)
```

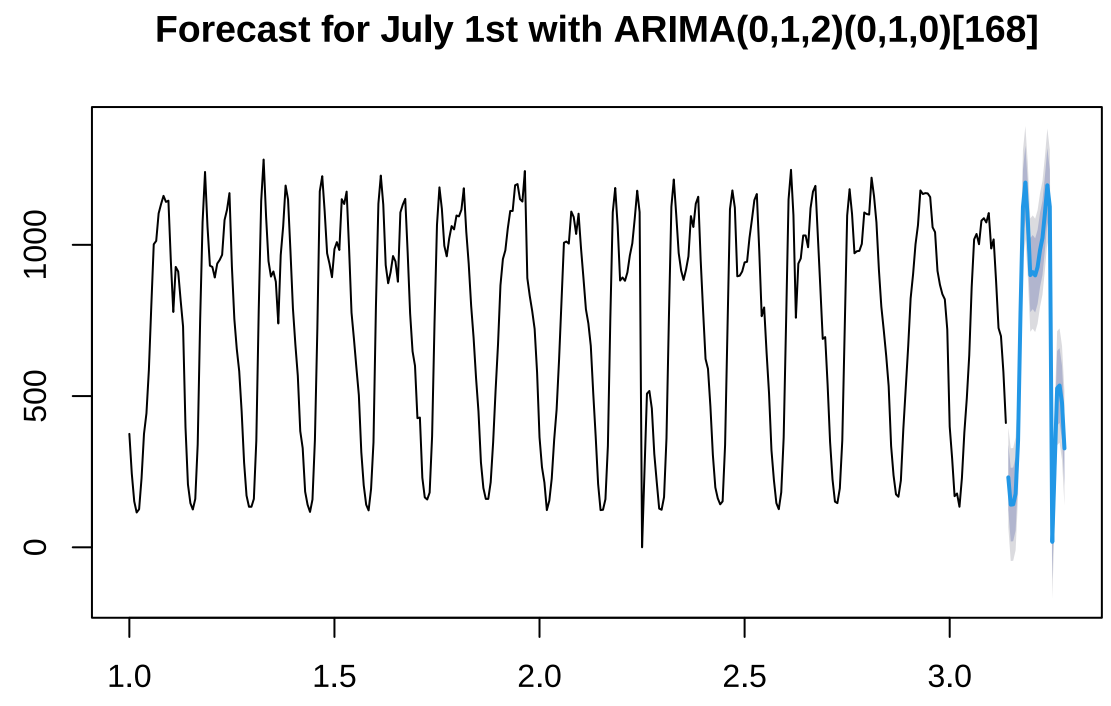
```r
# Use auto.arima
summary(auto_arima_model_weekly)
```

```
## Series: train_ts_weekly
## ARIMA(0,1,2)(0,1,0)[168]
##
```

```
## Coefficients:
##           ma1      ma2
##       -0.4741  -0.4853
## s.e.   0.0593   0.0586
##
## sigma^2 = 7081:  log likelihood = -1121.66
## AIC=2249.31   AICc=2249.44   BIC=2259.07
##
## Training set error measures:
##                    ME     RMSE      MAE  MPE MAPE      MASE      ACF1
## Training set 2.143998 60.97278 24.90656 -Inf  Inf 0.5210919 0.0387665
```

Question 5

```
# Forecast the next 24 hours
forecasted_values_week <- forecast(auto_arima_model_weekly, h=24)

# Plot the forecast
plot(forecasted_values_week, main="Forecast for July 1st with ARIMA(0,1,2)(0,1,0)[168]")
```



Question 6

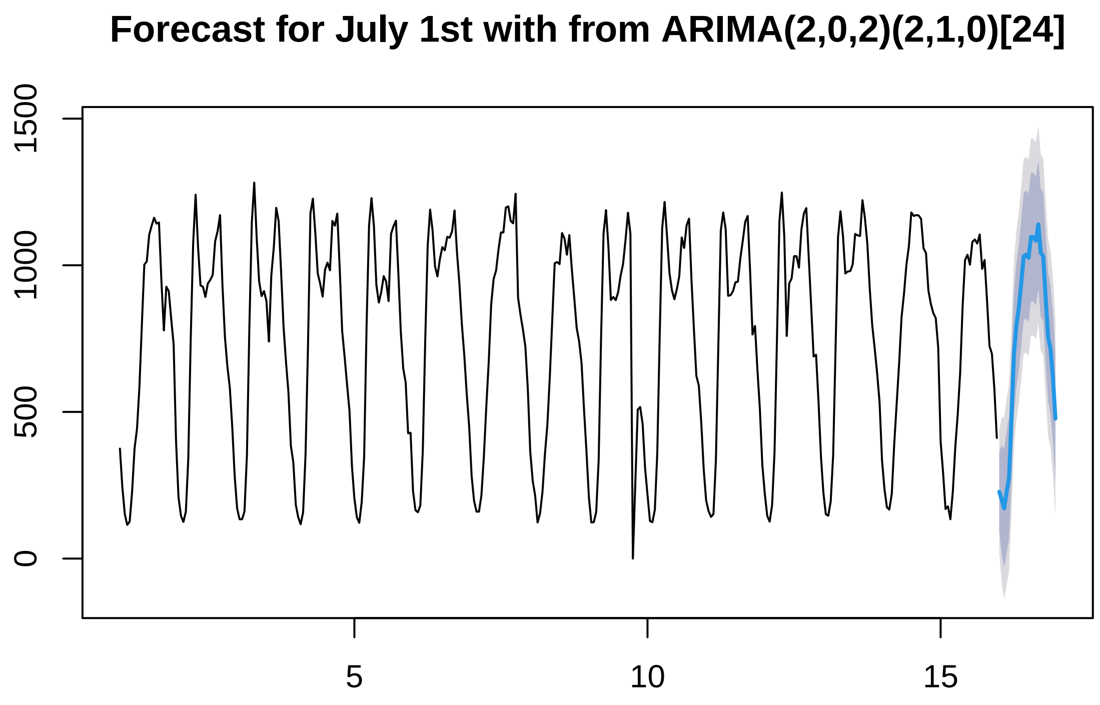
```r
# Create the time series object with frequency 24 for hourly data over a day
train_ts_daily <- ts(train$traffic_flow, frequency=24)
# Use auto.arima
auto_arima_model_daily <- auto.arima(train_ts_daily, seasonal=TRUE)
```

```r
summary(auto_arima_model_daily)
```

```
## Series: train_ts_daily
## ARIMA(2,0,2)(2,1,0)[24]
##
## Coefficients:
##          ar1      ar2      ma1      ma2     sar1     sar2
##       1.5965  -0.6895  -0.6772  -0.1650  -0.4151  -0.3254
## s.e.  0.0639   0.0521   0.0815   0.0694   0.0565   0.0535
##
## sigma^2 = 11243:  log likelihood = -2045.28
## AIC=4104.56   AICc=4104.9   BIC=4131.28
##
## Training set error measures:
##                    ME     RMSE      MAE  MPE MAPE      MASE        ACF1
## Training set 1.647241 101.5175 64.59878 -Inf  Inf 0.6546778 0.004301063
```

Question 7

```r
# Forecast the next 24 hours
forecasted_values_day <- forecast(auto_arima_model_daily, h=24)

# Plot the forecast
plot(forecasted_values_day, main="Forecast for July 1st with from ARIMA(2,0,2)(2,1,0)[24]")
```



Question 8

```r
# Extract forecasts for specific times
times <- c(8, 9, 17, 18)
forecast_week_values <- forecasted_values_week$mean[times]
forecast_day_values <- forecasted_values_day$mean[times]

# Display the forecasts
data.frame(
  Time = c("8:00 AM", "9:00 AM", "5:00 PM", "6:00 PM"),
  Forecast_Week = forecast_week_values,
  Forecast_Day = forecast_day_values
)
```

```
##      Time Forecast_Week Forecast_Day
## 1 8:00 AM      1205.979     794.1640
## 2 9:00 AM      1080.979     854.5382
## 3 5:00 PM      1196.979    1139.2999
## 4 6:00 PM      1125.979    1041.9611
```

```r
# extract the actual values for July 1st
actual_values <- c(traffic[368, "traffic_flow"],
                   traffic[369, "traffic_flow"],
                   traffic[377, "traffic_flow"],                                                  traffi
```

```r
# Transform the data type
actual_values <- as.numeric(actual_values)

# Calculate errors
mae_week <- mean(abs(forecast_week_values - actual_values))
mae_hour <- mean(abs(forecast_day_values - actual_values))

calc_rmse <- function(actual, predicted) {
  sqrt(mean((predicted - actual)^2))
}
rmse_week <- calc_rmse(actual_values, forecast_week_values)
rmse_hour <- calc_rmse(actual_values, forecast_day_values)

# Compare MAE and RMSE
data.frame(
  Model = c("auto_arima_model_weekly", "auto_arima_model_hourly"),
  MAE = c(mae_week, mae_hour),
  RMSE = c(rmse_week, rmse_hour)
)
```

```
##                     Model       MAE      RMSE
## 1 auto_arima_model_weekly  28.51072  33.92703
## 2 auto_arima_model_hourly 196.00918 257.59503
```

Based on the error, the first model in Q4, demonstrates a better predictive performance with a substantially lower Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) compared to the model in Q6. Lower values for these metrics indicate a closer fit to the actual observed data. Therefore, the auto_arima_model_weekly is preferred for its enhanced accuracy in forecasting.