McKinsey Analytics Online Hackathon

Bala Kesavan

18 November 2017, edited

# Summary

The contest requires prediction of traffic at four different junctions in a smart city. This analysis is an attempt to do so.

# Data exploration

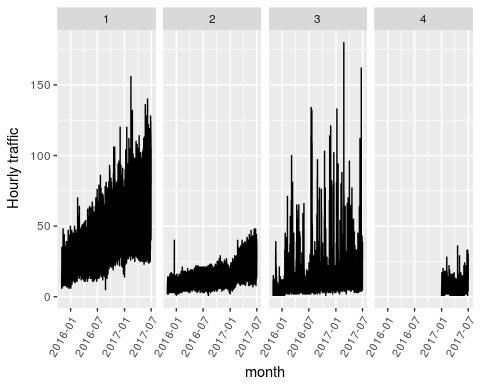
The graphical exploration conducted is reproduced below. Numerical checks were also done. Some of them are commented below.

library(forecast)  
library(tseries)  
library(ggplot2)  
  
setwd('/home/bala/Documents/Hackathon')  
McKTrainData = read.csv('train\_aWnotuB.csv')  
  
# head(McKTrainData)  
# tail(McKTrainData)  
# McKTrainData[1:25,]  
# str(McKTrainData)  
# table(McKTrainData$Junction)

## The four junctions need different prediction models

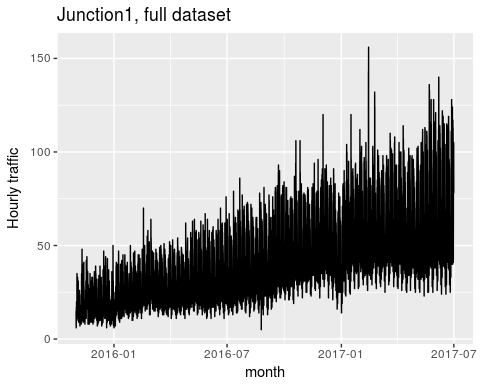
As seen below, the traffic patterns are very different.

McKTrainData$DateTime1 <- as.POSIXct(McKTrainData$DateTime, format="%Y-%m-%d %H:%M:%S")  
ggplot(McKTrainData, aes((DateTime1), Vehicles)) + geom\_line() + scale\_x\_datetime('month') +   
 ylab('Hourly traffic') + xlab("") + facet\_grid(.~Junction) + theme(axis.text.x = element\_text(angle=60, hjust = 1))

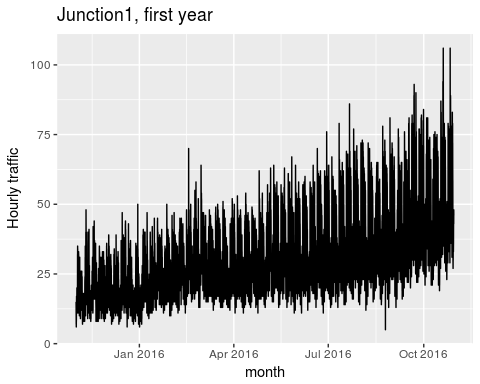


## Junction 1 is explored further in these graphs

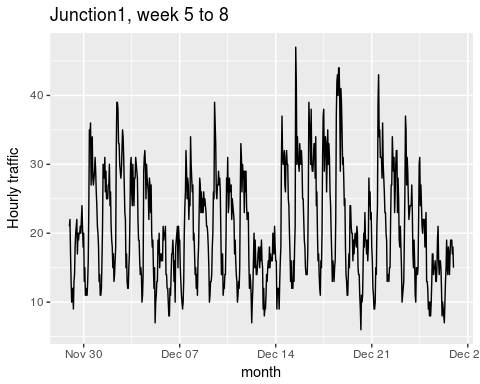
#subset data from junction 1 to probe/ understand dataset  
McKTrainData1 = McKTrainData[McKTrainData$Junction==1,]  
#junction1, full dataset  
ggplot(McKTrainData1, aes((DateTime1), Vehicles)) + geom\_line() + scale\_x\_datetime('month') +   
 ylab('Hourly traffic') + xlab("") + ggtitle("Junction1, full dataset")



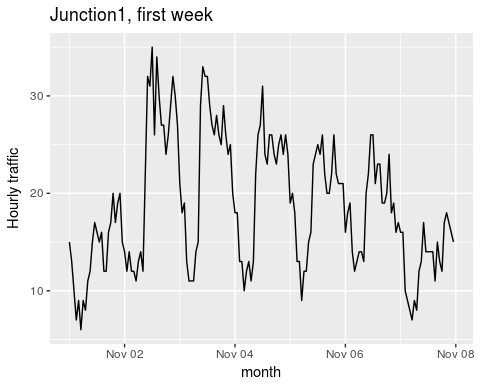
#junction1, first year  
ggplot(McKTrainData1[0:(24\*7\*52),], aes((DateTime1), Vehicles)) + geom\_line() + scale\_x\_datetime('month') +   
 ylab('Hourly traffic') + xlab("") + ggtitle("Junction1, first year")



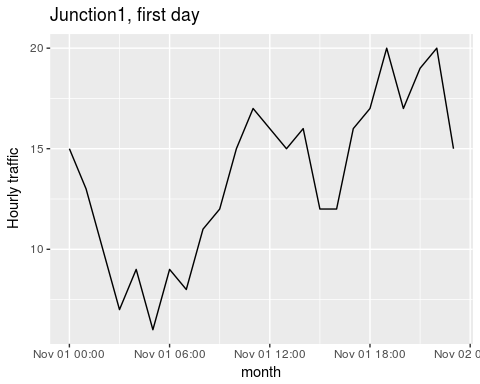
#junction1, second 4 weeks  
ggplot(McKTrainData1[(24\*7\*4):((24\*7\*4)+(24\*7\*4)),], aes((DateTime1), Vehicles)) + geom\_line() + scale\_x\_datetime('month') +   
 ylab('Hourly traffic') + xlab("") + ggtitle("Junction1, week 5 to 8")



#junction1, first week  
ggplot(McKTrainData1[0:(24\*7),], aes((DateTime1), Vehicles)) + geom\_line() + scale\_x\_datetime('month') +   
 ylab('Hourly traffic') + xlab("") + ggtitle("Junction1, first week")



#junction1, first day  
ggplot(McKTrainData1[0:(24),], aes((DateTime1), Vehicles)) + geom\_line() + scale\_x\_datetime('month') +   
 ylab('Hourly traffic') + xlab("") + ggtitle("Junction1, first day")

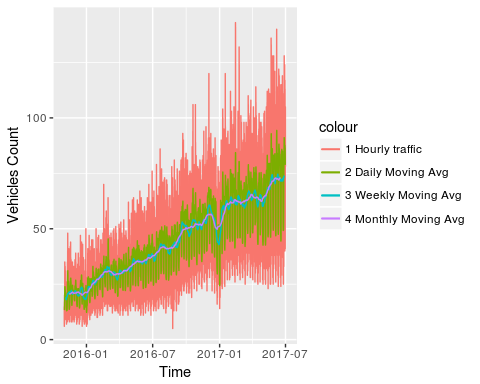


The daily and weekly graphs show clear seasonality - repetitive patterns at regular intervals.

## Moving averages

Calculating moving averages is the first step in deciphering signals embedded in time series data. Moving averages helps with identifying the lowest level of aggregation at which we can perform analysis and forecasting. We examine hourly, daily, weekly and monthly aggregations/ moving averages.

#outliers and missing values  
counts\_ts = ts(McKTrainData1[,c('Vehicles')])  
McKTrainData1$clean\_Vehicles = tsclean(counts\_ts)  
  
#smoothing with moving averages-  
McKTrainData1$Vehicles\_ma\_daily = ma(McKTrainData1$clean\_Vehicles, order = 24)  
McKTrainData1$Vehicles\_ma\_weekly = ma(McKTrainData1$clean\_Vehicles, order = 24\*7)  
McKTrainData1$Vehicles\_ma\_monthly = ma(McKTrainData1$clean\_Vehicles, order = 24\*7\*4)  
  
ggplot()+  
 geom\_line(data=McKTrainData1, aes(x=DateTime1, y=clean\_Vehicles, colour="1 Hourly traffic"))+  
 geom\_line(data=McKTrainData1, aes(x=DateTime1, y=Vehicles\_ma\_daily, colour="2 Daily Moving Avg"))+  
 geom\_line(data=McKTrainData1, aes(x=DateTime1, y=Vehicles\_ma\_weekly, colour="3 Weekly Moving Avg"))+  
 geom\_line(data=McKTrainData1, aes(x=DateTime1, y=Vehicles\_ma\_monthly, colour="4 Monthly Moving Avg"))+  
 ylab('Vehicles Count') + xlab('Time')

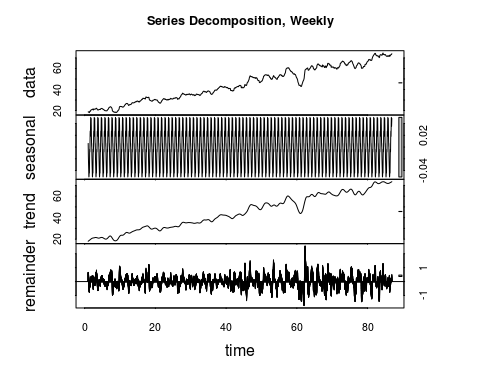


We see that the hourly and daily series are very volatile. The weekly series is reasonably stable and the monthly series provides no significant improvement in volatility. Series that are very volatile are hard to work with and require defining multiple seasonality levels. So, we will work with weekly moving averages.

## Decomposition

Now we convert the data into a time series and look for its key components, trend, seasonality and cycle, by decomposing the series.

Vehicles\_ma\_weekly = ts(na.omit(McKTrainData1$Vehicles\_ma\_weekly), frequency = (24\*7))  
#decomposition - weekly  
decomp\_weekly = stl(Vehicles\_ma\_weekly, s.window = 'periodic')  
plot(decomp\_weekly, main = "Series Decomposition, Weekly")

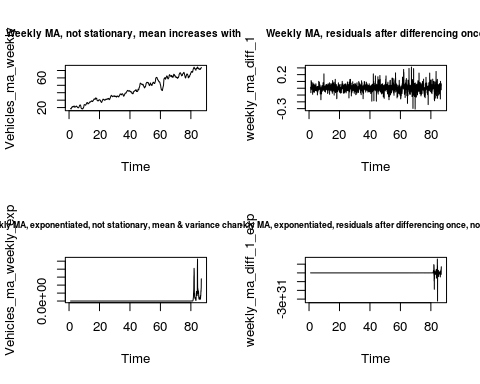


There is clearly seasonality in the weekly series. There is also an increasing trend. There is no apparent cycle, assuming the dip in Jan 2017 is not one.

## Stationary

Times Series modeling requires the Series to be stationary, which is defined as mean, variance and co-variance being time invariant. Working with the weekly moving average, we see there is an increasing trend for the mean. Hence we try differencing it and find that the residual (differenced) plot has time invariant mean and variance. No claim is being made about covariance at this stage.  
To illustrate stationarity, we also plot the weekly moving average series' exponential transformation and the residual from differencing once. The residual plot is not stationary and shows variance that depends on time.  
Just to confirm stationarity of the "differenced weekly moving average" series, we run the Augmented Dickey-Fuller Test.

par(mfrow=c(2,2))  
plot(Vehicles\_ma\_weekly, main = "Weekly MA, not stationary, mean increases with time", cex.main=0.75)  
  
weekly\_ma\_diff\_1 = diff(Vehicles\_ma\_weekly, differences = 1)  
plot(weekly\_ma\_diff\_1, main='Weekly MA, residuals after differencing once', cex.main=0.75)  
  
Vehicles\_ma\_weekly\_exp = ts(na.omit(exp(McKTrainData1$Vehicles\_ma\_weekly)), frequency = (24\*7))  
plot(Vehicles\_ma\_weekly\_exp, main = "Weekly MA, exponentiated, not stationary, mean & variance change with time", cex.main=0.65)  
  
weekly\_ma\_diff\_1\_exp = diff(exp(Vehicles\_ma\_weekly), differences = 1)  
plot(weekly\_ma\_diff\_1\_exp, main='Weekly MA, exponentiated, residuals after differencing once, not stationary', cex.main=0.65)



adf.test((weekly\_ma\_diff\_1))

## Warning in adf.test((weekly\_ma\_diff\_1)): p-value smaller than printed p-  
## value

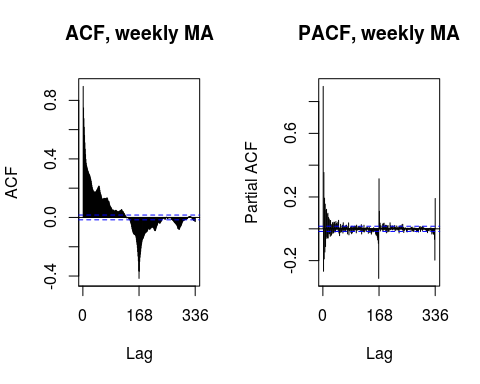
##   
## Augmented Dickey-Fuller Test  
##   
## data: (weekly\_ma\_diff\_1)  
## Dickey-Fuller = -14.342, Lag order = 24, p-value = 0.01  
## alternative hypothesis: stationary

Due to the low p-value in the Augmented Dickey-Fuller Test, the null hypothesis of non-stationarity has to be rejected.

## Looking for patterns

The idea behind time series analysis and forecasting is that any reading, traffic in this case, depends on prior readings. This relationship to prior readings has two components. Auto Regressive (AR) - directly the influence of the prior readings. Moving Average (MA) - the influence of the prior periods' errors. Using the Auto Correlation Factor (ACF) and the Partial Auto Correlation Factor (PACF) plots of the differenced weekly moving average series, we see that there AR (parameter p in an ARIMA model) and MA (parameter q) components to be discovered in the residuals. The ACF plot shows auto correlation with prior lags. Depending on how this is addressed, there may remain a dependence between the error terms, as seen in the PACF plot.

par(mfrow=c(1,2))  
Acf(weekly\_ma\_diff\_1, main='ACF, weekly MA')  
Pacf(weekly\_ma\_diff\_1, main='PACF, weekly MA')



## ARIMA model training

We leverage the algorithm's ability to discover the optimal Auto Regression(p and P), Integration(d and D), Moving Averages (q and Q) parameters. The P, D and Q being the seasonal parameters. We pick the model with the best AIC, BIC scores.  
For simplicity, we model de-seasonal series by subtracting the seasonal components from the original series.

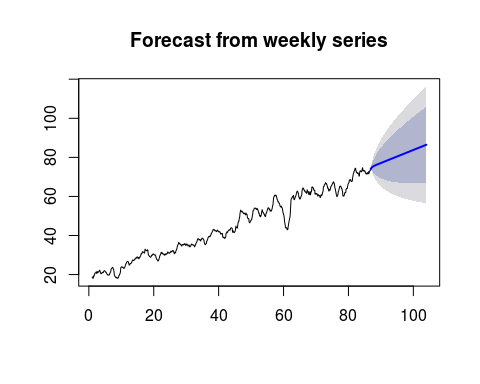
deseasonal\_cnt\_weekly = seasadj(decomp\_weekly)  
fit\_w\_seasonality\_weekly = auto.arima(deseasonal\_cnt\_weekly, seasonal = T)  
fit\_w\_seasonality\_weekly

## Series: deseasonal\_cnt\_weekly   
## ARIMA(5,1,4) with drift   
##   
## Coefficients:  
## ar1 ar2 ar3 ar4 ar5 ma1 ma2 ma3  
## 0.9605 0.7027 -0.6760 0.0485 -0.0405 0.5003 -1.2140 -0.4566  
## s.e. 0.0653 0.0516 0.0764 0.0286 0.0170 0.0649 0.0628 0.0583  
## ma4 drift  
## 0.2635 0.0040  
## s.e. 0.0560 0.0024  
##   
## sigma^2 estimated as 0.0002161: log likelihood=40400.72  
## AIC=-80779.44 AICc=-80779.42 BIC=-80696.09

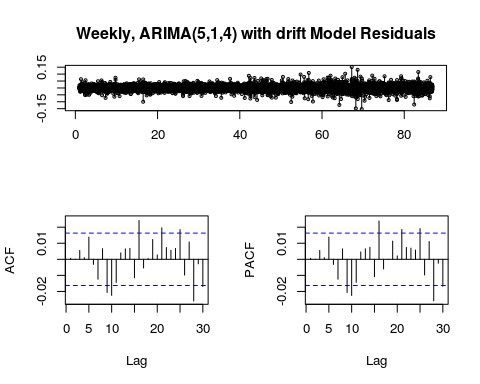
## Evaluating the model

We forecast, plot the forecast and examine the residuals, Auto Correlation Function and Partial Auto Correlation Function for all the models. The residuals show no pattern and are in a tight band around zero. This is good. The ACF and PACF plots show that patterns in the data are substantially accounted for, as seen in how much smaller the bars are and how few of them exceed the blue significance zone. Although, it appears that there is more pattern information left at lag 9, 10 etc. that can be accounted for in the ARIMA model parameters.

seas\_fcast\_weekly <- forecast(fit\_w\_seasonality\_weekly, h=24\*30\*4)  
plot(seas\_fcast\_weekly, main="Forecast from weekly series")



tsdisplay(residuals(seas\_fcast\_weekly), lag.max = 30, main = 'Weekly, ARIMA(5,1,4) with drift Model Residuals')



## Conclusion

This document is meant to demonstrate approach and the same can be extended to build statistically sound, well tuned models for all four junctions.