Forecasting

RGV

8 December 2019

# Arima for forecasting

### Loading the relevant packages and data

library(forecast)  
library(lubridate)

##   
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':  
##   
## date

library(tseries)  
library(fpp)

## Loading required package: fma

## Loading required package: expsmooth

## Loading required package: lmtest

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

base\_data<-read.csv('D:/Freelancer\_questions/kevin/1134769745\_daily\_with\_workdays\_27.csv')  
colnames(base\_data)<-c("Date","Temperature","Humidity","Windspeed","Count","All\_workdays")  
base\_data<-base\_data[order(as.Date(base\_data$Date, format="%d/%m/%Y")),]  
base\_data<-base\_data[order(as.Date(base\_data$Date, format="%d/%m/%Y")),]  
base\_data$All\_workdays<-as.factor(base\_data$All\_workdays)  
head(base\_data)

## Date Temperature Humidity Windspeed Count All\_workdays  
## 1 07/07/2016 21.55958 70.04167 2.291667 184 1  
## 2 08/07/2016 22.31042 74.08333 2.666667 339 1  
## 3 09/07/2016 22.95125 75.45833 2.250000 409 0  
## 4 10/07/2016 23.32250 74.41667 2.708333 333 0  
## 5 11/07/2016 22.69667 68.41667 2.500000 344 1  
## 6 12/07/2016 22.59608 67.95833 2.500000 522 1

### Dividing the data into training and test set

#ARIMA programming starts  
## 75% of the sample size  
smp\_size <- floor(0.95 \* nrow(base\_data))  
print(smp\_size)

## [1] 1033

## set the seed to make your partition reproducible  
set.seed(123)  
train\_ind <- sample(seq\_len(nrow(base\_data)), size = smp\_size)  
  
train <- base\_data[1:smp\_size, ]  
test <- base\_data[smp\_size+1:nrow(base\_data), ]  
test<-na.omit(test)

### Create matrix of numeric predictors

xreg <- cbind(as\_workday=model.matrix(~train$All\_workdays),   
 Temp=train$Temperature,  
 Humidity=train$Humidity,  
 Windspeed=train$Windspeed  
 )  
  
# Remove intercept  
xreg <- xreg[,-1]  
  
# Rename columns  
colnames(xreg) <- c("All\_workdays","Temp","Humidity","Windspeed")  
  
#creating the same for the test data  
  
xreg1 <- cbind(as\_workday=model.matrix(~test$All\_workdays),   
 Temp=test$Temperature,  
 Humidity=test$Humidity,  
 Windspeed=test$Windspeed  
)  
  
# Remove intercept  
xreg1 <- xreg1[,-1]  
  
# Rename columns  
colnames(xreg1) <- c("All\_workdays","Temp","Humidity","Windspeed")

### Creating a time series variable for the traning data for arima forecasting

Count <- ts(train$Count, start=c(2016.6),frequency=365)

Inference : As the data is for each day the frequency will be 365 and the start date is 2016-7-7

### Fitting ARIMA model with seasonality

modArima <- auto.arima(Count, xreg=xreg)  
modArima

## Series: Count   
## Regression with ARIMA(1,1,1) errors   
##   
## Coefficients:  
## ar1 ma1 All\_workdays Temp Humidity Windspeed  
## 0.1945 -0.8571 36.8114 -0.8123 -1.2827 -24.4085  
## s.e. 0.0380 0.0198 7.6259 2.0231 0.2886 6.3385  
##   
## sigma^2 estimated as 13801: log likelihood=-6380.6  
## AIC=12775.2 AICc=12775.31 BIC=12809.78

Forecasted\_values<-forecast(modArima,nrow(test),xreg=xreg1)  
  
Final\_forecasted\_values<-Forecasted\_values$mean  
length(Final\_forecasted\_values)

## [1] 55

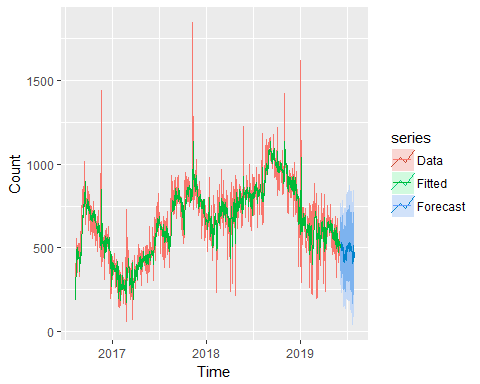
### calculating the MSE in the test dataset

mean((test$Count - Final\_forecasted\_values)^2)

## [1] 51860.74

### plotting forecasted values before seasonality removal

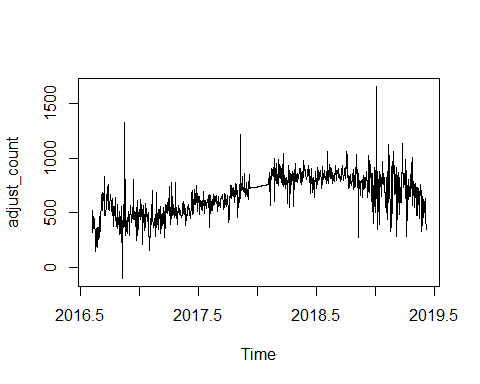
library(ggplot2)  
forecast(modArima,nrow(test),xreg=xreg1) -> fc  
autoplot(Count, series="Data") +   
autolayer(fc, series="Forecast") +   
autolayer(fitted(fc), series="Fitted")



### Fitting ARIMA without seasonality

### Deseasonaling the dataset and plotting

decompose\_data = decompose(Count, "additive")  
adjust\_count = Count - decompose\_data$seasonal  
plot(adjust\_count)

 ### Find ARIMAX model of deseasoned data

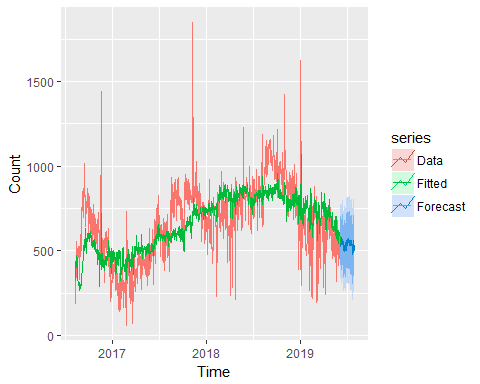
modArima\_desea <- auto.arima(adjust\_count, xreg=xreg)  
modArima\_desea

## Series: adjust\_count   
## Regression with ARIMA(3,1,1) errors   
##   
## Coefficients:  
## ar1 ar2 ar3 ma1 All\_workdays Temp Humidity  
## 0.2376 0.0552 -0.0288 -0.9219 33.6719 -2.7078 -0.8318  
## s.e. 0.0360 0.0346 0.0343 0.0184 7.0878 1.7676 0.2657  
## Windspeed  
## -16.5851  
## s.e. 5.7804  
##   
## sigma^2 estimated as 11553: log likelihood=-6288.04  
## AIC=12594.09 AICc=12594.26 BIC=12638.54

Forecasted\_values\_des<-forecast(modArima\_desea,nrow(test),xreg=xreg1)  
Final\_forecasted\_values\_des<-Forecasted\_values\_des$mean

### plotting forecasted values after seasonality removal

library(ggplot2)  
forecast(modArima\_desea,nrow(test),xreg=xreg1) -> fc1  
autoplot(Count, series="Data") +   
autolayer(fc1, series="Forecast") +   
autolayer(fitted(fc1), series="Fitted")



### Mean square Error component

mean((test$Count - Final\_forecasted\_values\_des)^2)

## [1] 36754.14

### Dynamic regression by taking lag variables of output as well as input with lag 1,2

library(dplyr)

## Warning: package 'dplyr' was built under R version 3.5.1

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:lubridate':  
##   
## intersect, setdiff, union

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

x<-train[order(as.Date(train$Date, format="%d/%m/%Y")),]  
  
train\_aug <- x %>%  
 mutate(count\_lag1 = lag(Count, n = 1, order\_by = Date),  
 count\_lag2 = lag(Count, n = 2, order\_by = Date),  
 temp\_lag1 = lag(Temperature, n = 1, order\_by = Date),  
 temp\_lag2 = lag(Temperature, n = 2, order\_by = Date),  
 Hum\_lag1 = lag(Humidity, n = 1, order\_by = Date),  
 Hum\_lag2 = lag(Humidity, n = 2, order\_by = Date),  
 Wind\_lag1 = lag(Windspeed, n = 1, order\_by = Date),  
 Wind\_lag2 = lag(Windspeed, n = 2, order\_by = Date)  
 )  
  
x1<-test[order(as.Date(test$Date, format="%d/%m/%Y")),]  
  
test\_aug <- x1 %>%  
 mutate(count\_lag1 = lag(Count, n = 1, order\_by = Date),  
 count\_lag2 = lag(Count, n = 2, order\_by = Date),  
 temp\_lag1 = lag(Temperature, n = 1, order\_by = Date),  
 temp\_lag2 = lag(Temperature, n = 2, order\_by = Date),  
 Hum\_lag1 = lag(Humidity, n = 1, order\_by = Date),  
 Hum\_lag2 = lag(Humidity, n = 2, order\_by = Date),  
 Wind\_lag1 = lag(Windspeed, n = 1, order\_by = Date),  
 Wind\_lag2 = lag(Windspeed, n = 2, order\_by = Date)  
 )

### OLS regression using the Dynamic lagged variables

my\_lm <- lm(Count ~ count\_lag1 + count\_lag2 + temp\_lag1 + temp\_lag2 +Hum\_lag1+Hum\_lag2+Wind\_lag1+Wind\_lag2,data = train\_aug[3:nrow(train\_aug), ])  
summary(my\_lm)

##   
## Call:  
## lm(formula = Count ~ count\_lag1 + count\_lag2 + temp\_lag1 + temp\_lag2 +   
## Hum\_lag1 + Hum\_lag2 + Wind\_lag1 + Wind\_lag2, data = train\_aug[3:nrow(train\_aug),   
## ])  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -614.77 -130.09 9.72 132.51 1165.74   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 294.09214 47.59342 6.179 9.29e-10 \*\*\*  
## count\_lag1 -0.24892 0.03336 -7.463 1.81e-13 \*\*\*  
## count\_lag2 -0.11772 0.03275 -3.595 0.00034 \*\*\*  
## temp\_lag1 20.62815 1.78986 11.525 < 2e-16 \*\*\*  
## temp\_lag2 8.43898 1.90639 4.427 1.06e-05 \*\*\*  
## Hum\_lag1 0.48761 0.38250 1.275 0.20267   
## Hum\_lag2 0.52731 0.38248 1.379 0.16829   
## Wind\_lag1 17.31211 8.33838 2.076 0.03813 \*   
## Wind\_lag2 -6.19091 8.35632 -0.741 0.45895   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 193.3 on 1020 degrees of freedom  
## (2 observations deleted due to missingness)  
## Multiple R-squared: 0.2286, Adjusted R-squared: 0.2225   
## F-statistic: 37.78 on 8 and 1020 DF, p-value: < 2.2e-16

Inference: Keeping only the significant variables where the P value is <0.05 and removing the other variables

### OLS regression recreated with keeping only the significant variables

My\_lm\_final <-lm(Count ~ count\_lag1 + count\_lag2 + temp\_lag1 + temp\_lag2 , data = train\_aug[3:nrow(train\_aug), ])  
summary(My\_lm\_final)

##   
## Call:  
## lm(formula = Count ~ count\_lag1 + count\_lag2 + temp\_lag1 + temp\_lag2,   
## data = train\_aug[3:nrow(train\_aug), ])  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -594.78 -131.38 5.66 134.94 1135.19   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 378.78521 29.68109 12.762 < 2e-16 \*\*\*  
## count\_lag1 -0.27122 0.03221 -8.420 < 2e-16 \*\*\*  
## count\_lag2 -0.11323 0.03173 -3.568 0.000376 \*\*\*  
## temp\_lag1 21.15650 1.71722 12.320 < 2e-16 \*\*\*  
## temp\_lag2 8.41413 1.83701 4.580 5.21e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 193.8 on 1024 degrees of freedom  
## (2 observations deleted due to missingness)  
## Multiple R-squared: 0.222, Adjusted R-squared: 0.2189   
## F-statistic: 73.04 on 4 and 1024 DF, p-value: < 2.2e-16

### predicting the same on the test data to calculate MSE

predicted\_Dynm<-predict(My\_lm\_final,newdata = test\_aug)  
  
test\_aug$Predicted<-predicted\_Dynm  
test\_aug\_1<-na.omit(test\_aug)  
# Mean Square error for the dynamic regression#  
mean((test\_aug\_1$Count - test\_aug\_1$Predicted)^2)

## [1] 28639.19

### ploting predicted vs actual

plot(test\_aug\_1$Count,test\_aug\_1$Predicted,  
 xlab="predicted",ylab="actual")  
 abline(a=0,b=1)

