Regressors of Diwali on Industrial Production of India

Diwali is the most important festival of India and the timing of its distorts the monthly time-series of industrial production heavily. Generally Diwali is celebrated in the month of October according to Gregorian Calendar but that is not fixed and depending on which month it is celebrated the industrial production index also changes. The standard software packages for seasonal adjustment, `X-12-ARIMA` and `X-13-ARIMA-SEATS` (developed by the U.S. Census Bureau) or Tramo Seats (developed by the Bank of Spain) have a built-in adjustment procedure for Easter holiday, but not for Diwali. However, all packages allow for the inclusion of user defined variables, and the Chinese New Year can be modeled as such. `seasonal` (Sax and Eddelbuettel 2018) is an interface to X-13ARIMA-SEATS.

```
rm(list = ls())
library(seasonal)
library(VedicDateTime)
library(insol)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

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

```
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
data(seasonal)
data(holiday)
```

Considering Industrial Production of India after 2000

```
industrial_prod <- window(iip, start = 2000)
```

Convert Gregorian dates of Diwali holidays to Vedic calendar dates

```
get_vedic_date<- function(julian_day, place) {

masa_num <- VedicDateTime::masa(julian_day, place)
vikram_samvatsara = VedicDateTime::elapsed_year(julian_day, masa_num)[5]
tithi_ = tithi(julian_day, place)[1]
masa_ = masa(julian_day, place)[1]
vedic_dates = paste(as.character(vikram_samvatsara),"-",as.character(masa_),"-",as.character(tithi_), s
return(vedic_dates)
}
```

Converted Diwali dates to vedic datetime compatible date-times

```
date<- seasonal::diwali
date <- as.POSIXct.Date(date)
julianday <- insol::JD(date)


place <- c(15.34, 75.13, +5.5) #Latitude, Longitude and timezone of the location
diwali_vedic_calendar = c()

for (i in 1:length(julianday))
{
  diwali_vedic_calendar = c(diwali_vedic_calendar, get_vedic_date(julianday[i], place))
}
```

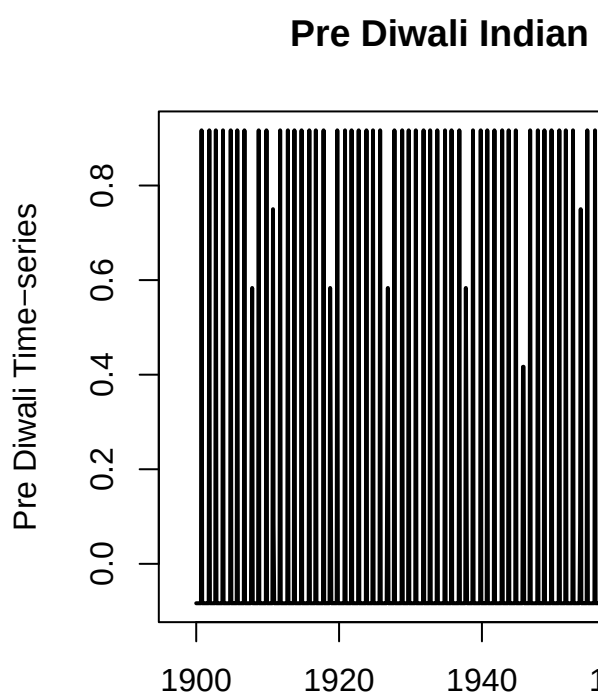
Generate time-series based on 'genhol()' function using dates of Diwali as input

Generate time-series based on Vedic calendar dates of Diwali

```
pre_diwali_ts_vedic <- genhol(as.Date(diwali_vedic_calendar), start = -6, end = -1, center = "mean")
post_diwali_ts_vedic <- genhol(as.Date(diwali_vedic_calendar), start = 0, end = 6, center = "mean")
```

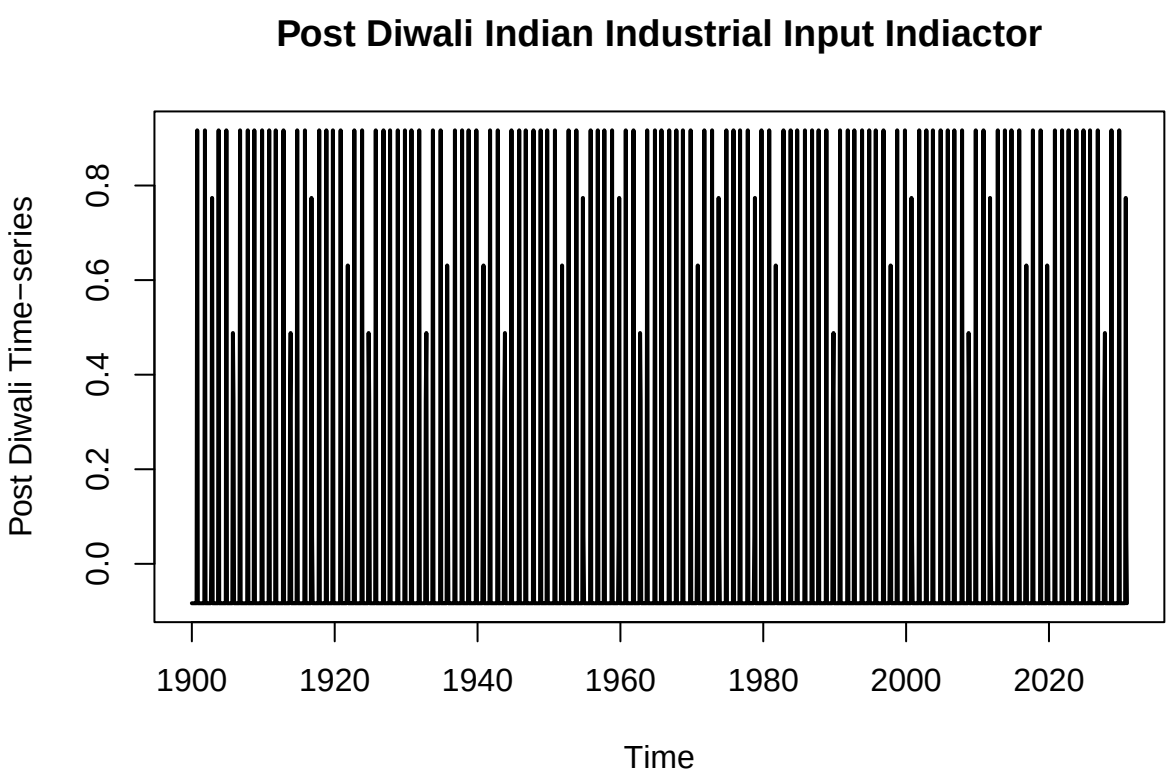
`pre_diwali_ts` and `post_diwali_ts` both are of time-series class object and represent 2 time-series to include pre and post festival for better seasonal adjustment.

```
pre_diwali_ts<- genhol(seasonal::diwali, start = -6, end = -1, center = "mean")
post_diwali_ts<- genhol(seasonal::diwali, start = 0, end = 6, center = "mean")
ts.plot(pre_diwali_ts,lwd = c(2, 1),  gpars=list(xlab="Time", ylab="Pre Diwali Time-series", main = "Pr
```



Generate time-series with Gregorian calendar system for Diwali

```
ts.plot(post_diwali_ts,lwd = c(2, 1), gpars=list(xlab="Time", ylab="Post Diwali Time-series", main = "P
```



Including user defined regressors

Converting India Industrial Output time-series object to Vedic Calendar system

Converting 'seasonal::iip' ts object to Vedic calendar based time-series object to model seasonality

```
iip_data <- data.frame(Y=as.matrix(seasonal::iip), date=as.Date(zoo::as.yearmon(time(seasonal::iip))))
date<- iip_data$date
date <- as.POSIXct.Date(date)
julianday_iip <- insol::JD(date)
place <- c(15.34, 75.13, +5.5) #Latitude, Longitude and timezone of the location
iip_vedic_calendar = c()
for (i in 1:length(julianday_iip))
{
iip_vedic_calendar = c(iip_vedic_calendar, get_vedic_date(julianday_iip[i], place))
}
iip_data$date <- iip_vedic_calendar
iip_vedic_data_ts <- ts(iip_data$Y,start=c(2061,2))
```

The `seasonal` package allows to add user-defined regressors to remove seasonality from a time-series. Here `pre_diwali_ts` and `post_diwali_ts` are added in the main seasonal adjustment. `X-13ARIMA-SEATS` is used to adjust for the festival seasonal component.

```
m1 <- seas(industrial_prod, xreg = cbind(pre_diwali_ts, post_diwali_ts), regression.usertype = "holiday
```

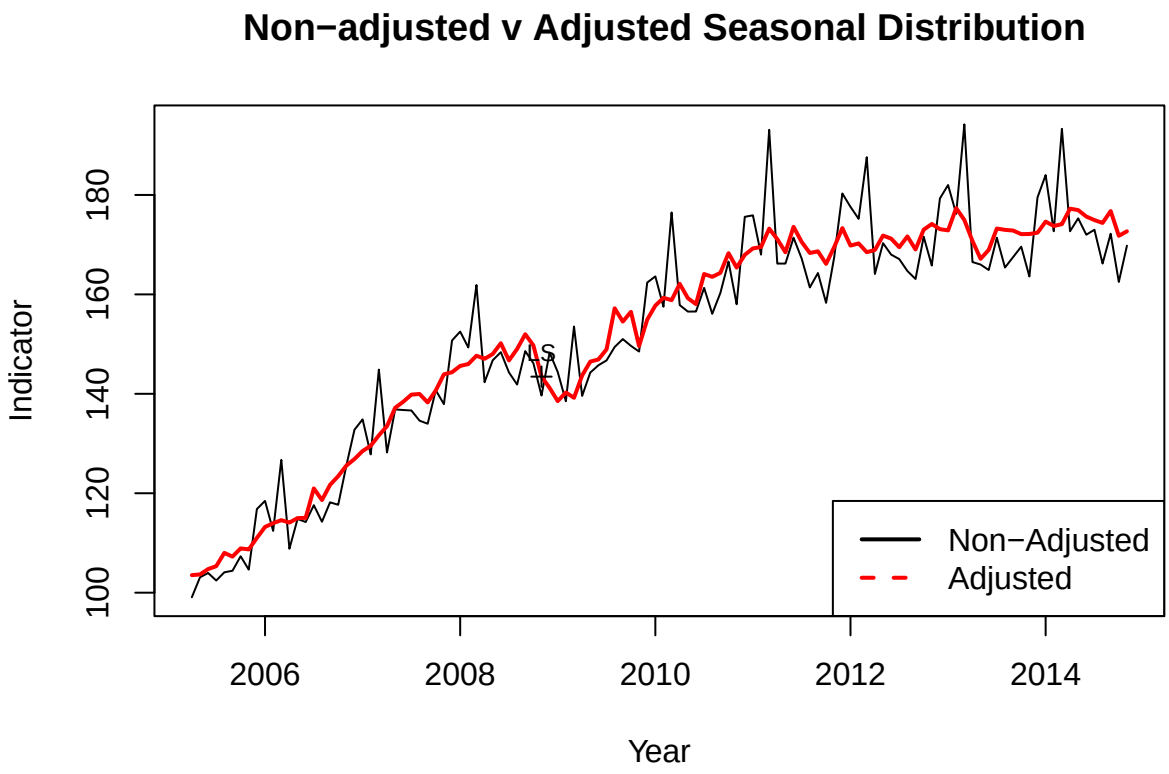
`xreg` adds the user-defined regressors and `x11` is chosen as the decomposition effect.

```r
summary(m1)
```

```
##
## Call:
## seas(x = industrial_prod, xreg = cbind(pre_diwali_ts, post_diwali_ts),
##     regression.usertype = "holiday", x11 = list())
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## xreg1             -0.0034498  0.0091813  -0.376  0.70711
## xreg2             -0.0383834  0.0081358  -4.718 2.38e-06 ***
## Weekday            0.0012087  0.0004275   2.827  0.00469 **
## Constant          -0.0014297  0.0003227  -4.431 9.39e-06 ***
## LS2008.Nov        -0.0738919  0.0160213  -4.612 3.99e-06 ***
## MA-Nonseasonal-01  0.4328295  0.0785625   5.509 3.60e-08 ***
## MA-Seasonal-12     0.9995753  0.0785461  12.726  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## X11 adj.  ARIMA: (0 1 1)(0 1 1)  Obs.: 116  Transform: log
## AICc: 543.2, BIC: 562.7  QS (no seasonality in final):    0
## Box-Ljung (no autocorr.): 31.53   Shapiro (normality): 0.9894
```

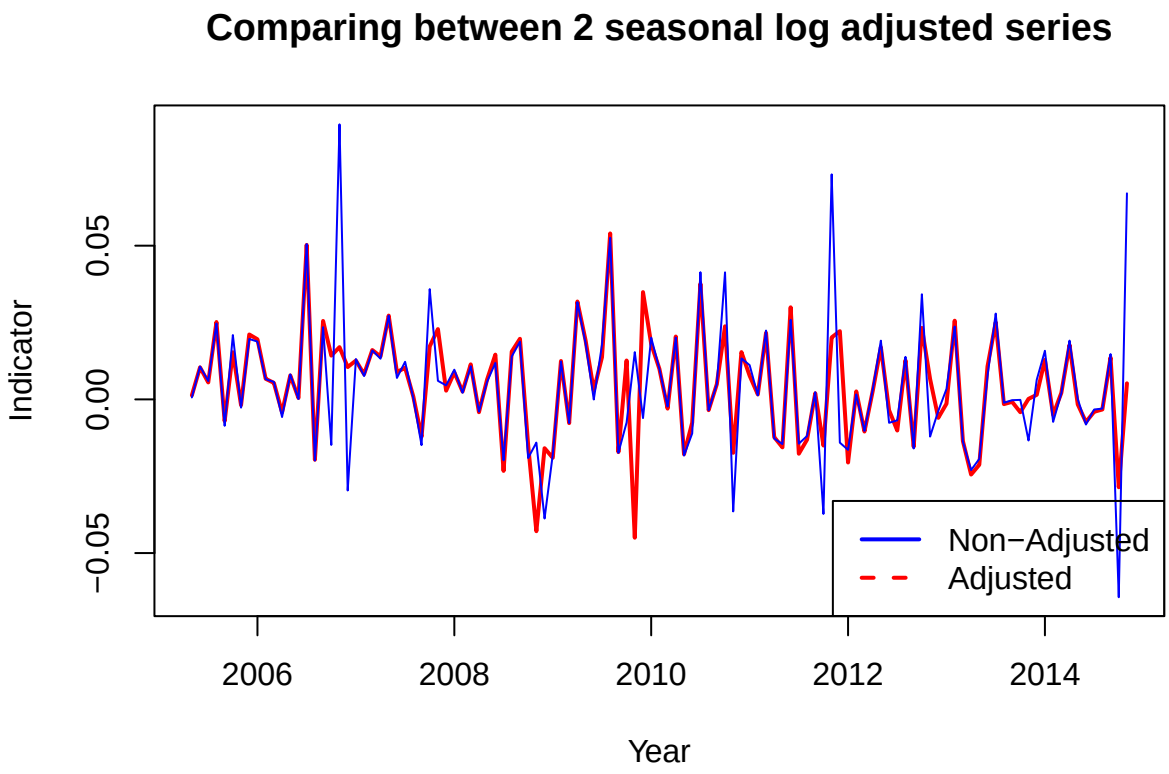
The seasonal co-efficient shows minor decline during pre and post Diwali season across the time-series. In the below non-adjusted v adjusted seasonal plot it can be observed that seasonal adjustment based on Diwali season removes distortion from the time-series.

```r
plot(m1, xlab="Year", ylab="Indicator", main = " Non-adjusted v Adjusted Seasonal Distribution")
lines(iip_vedic_data_ts, col="red")
legend(x = "bottomright",
       legend = c("Non-Adjusted", "Adjusted"),
       lty = c(1, 2),
       col = c("black", "red"),
       lwd = 2)
```



Comparing the series

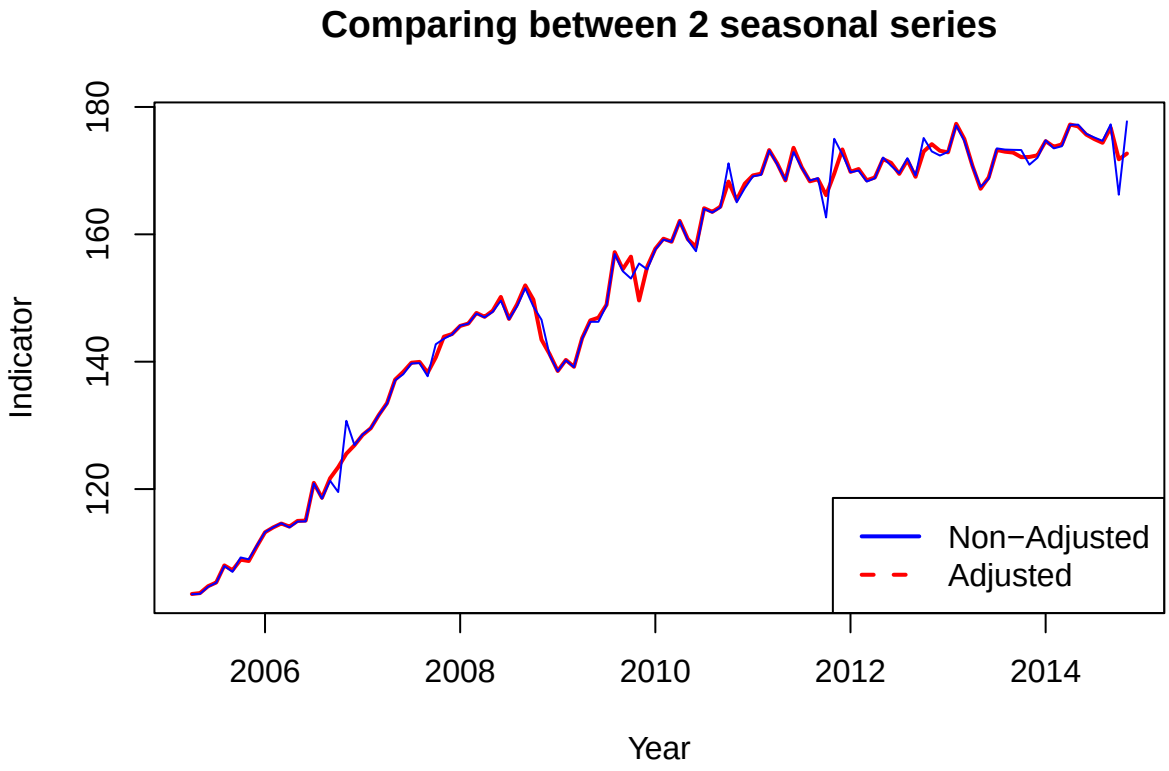
```
m2 <- seas(x = iip, x11 = list(), regression.variables = c("td1coef", "ls2008.Nov"),
           arima.model = "(0 1 1)(0 1 1)", regression.aictest = NULL, outlier = NULL,
           transform.function = "log")
ts.plot(diff(log(cbind(final(m1), final(m2)))), col = c("red", "blue"), lwd = c(2, 1),  gpars=list(xlab=
legend(x = "bottomright",
       legend = c("Non-Adjusted", "Adjusted"),
       lty = c(1, 2),
       col = c("blue", "red"),
       lwd = 2)
```



In the above chart, non-adjusted(blue) vs adjusted(red) seasonal plot clearly shows the amount of distortion present in the series.

The below chart also indicated a level of distortion present for industrial output due to Diwali festival.

```
ts.plot(final(m1), final(m2), col = c("red", "blue"), lwd = c(2, 1), gpars=list(xlab="Year", ylab="Indi
legend(x = "bottomright",
       legend = c("Non-Adjusted", "Adjusted"),
       lty = c(1, 2),
       col = c("blue", "red"),
       lwd = 2)
```



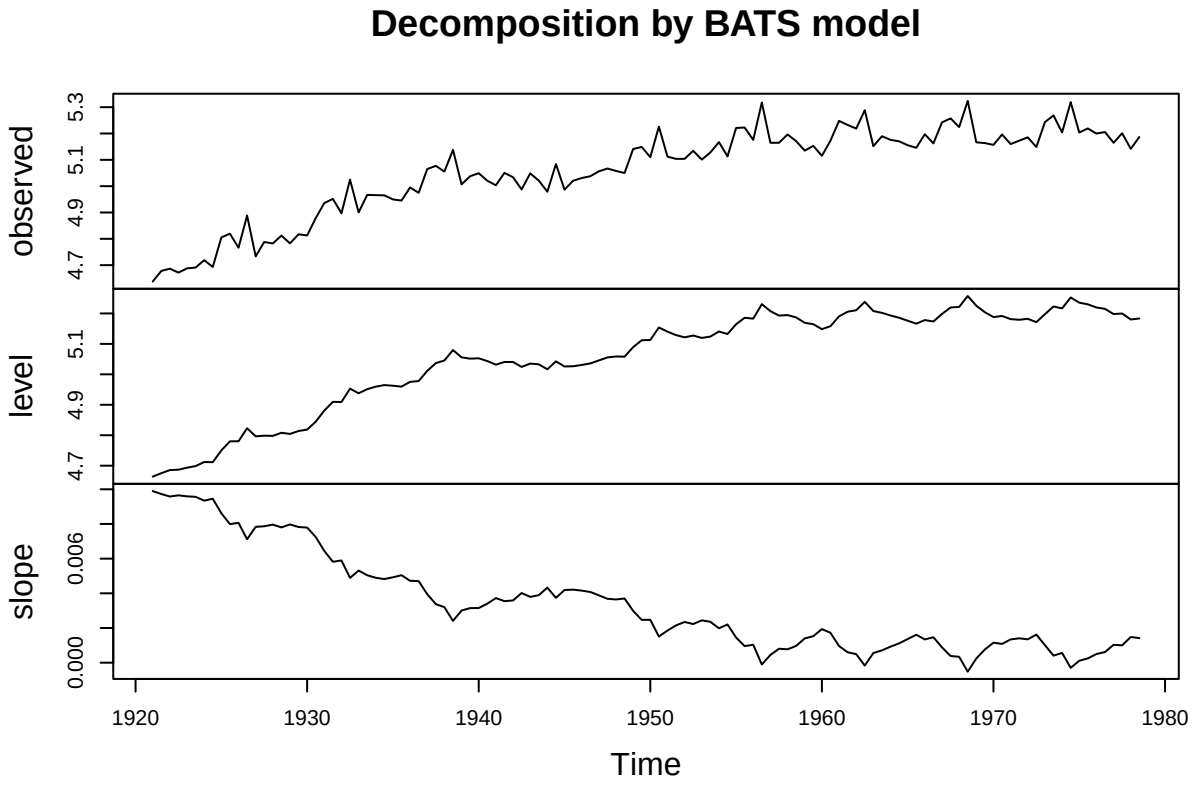
Multi-seasonality decomposition for India industrial output

(**forecast?**) library was used to identify multiple seasonality present in the data-set and understand the difference between them.

Seasonality decomposition for Industrial Output for Vedic Calendar

The start of the series is 1921, according to Vedic calendar system which is equivalent to 2000

```
iip_tbats_vedic <- forecast::msts(iip_vedic_data_ts, start=1921, seasonal.periods = c(2))
fit_vedic <- forecast::tbats(iip_tbats_vedic)
plot(fit_vedic)
```



Sax, Christoph, and Dirk Eddelbuettel. 2018. “Seasonal Adjustment by {x-13ARIMA-SEATS} in {r}” 87. https://doi.org/10.18637/jss.v087.i11.