Series Temporales

Mini-tutorial on the use of zoo and xts packages

Jakub Maciążek

Exercise description

Make an .rmd with a mini-tutorial on the use of the zoo and xts packages, applying them to a time series (dataset with data from Spain) that you search on the web about electricity consumption, product consumption, etc.

Data source

Following tutorial will be based on data related to electricy consumption, specifically daily energy SPOT price, from the years 2014 - 2018.

Dataset source: https://www.kaggle.com/datasets/manualrg/spanish-electricity-market-demand-gen-price.

```
#Load data

spain_energy_market <- read.csv("S:/0_Universidad_de_Malaga/MI_Ingenieria_y_ciencia_de_datos/Estatistica
spain_SPOT_prices <- filter(spain_energy_market, name=='Precio mercado SPOT Diario ESP')
head(spain_SPOT_prices)
##              datetime  id                           name geoid geoname
## 1 2014-01-01 23:00:00 600 Precio mercado SPOT Diario ESP     3 EspaÃ±a
## 2 2014-01-02 23:00:00 600 Precio mercado SPOT Diario ESP     3 EspaÃ±a
## 3 2014-01-03 23:00:00 600 Precio mercado SPOT Diario ESP     3 EspaÃ±a
## 4 2014-01-04 23:00:00 600 Precio mercado SPOT Diario ESP     3 EspaÃ±a
## 5 2014-01-05 23:00:00 600 Precio mercado SPOT Diario ESP     3 EspaÃ±a
## 6 2014-01-06 23:00:00 600 Precio mercado SPOT Diario ESP     3 EspaÃ±a
##       value
## 1 25.280833
## 2 39.924167
## 3  4.992083
## 4  4.091667
## 5 13.587500
## 6 47.885417

keeps <- c("datetime", "value")
data <- spain_SPOT_prices[keeps]
head(data)
##              datetime     value
## 1 2014-01-01 23:00:00 25.280833
## 2 2014-01-02 23:00:00 39.924167
## 3 2014-01-03 23:00:00  4.992083
```

```
## 4 2014-01-04 23:00:00  4.091667
## 5 2014-01-05 23:00:00 13.587500
## 6 2014-01-06 23:00:00 47.885417
```

Brief package introduction

xts - eXtensible Time Series

zoo object - index + matrix -> observation in time

Functionalities

First basic functionalities are presented. Those include creation of xts objects, and conversion back to raw data frame.

Basics

```
#Import package
library(xts)
## Ładowanie wymaganego pakietu: zoo
##
## Dołączanie pakietu: 'zoo'
## Następujące obiekty zostały zakryte z 'package:base':
##
##     as.Date, as.Date.numeric
##
## ########################## Warning from 'xts' package ##########################
## #                                                                              #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                             #
## #                                                                              #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop            #
## # dplyr from breaking base R's lag() function.                                 #
## #                                                                              #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                              #
## ################################################################################
##
## Dołączanie pakietu: 'xts'
## Następujące obiekty zostały zakryte z 'package:dplyr':
##
##     first, last

# Construct xts
x <- xts(data$value, order.by = as.Date(data$datetime))
head(x)
```

```
##                  [,1]
## 2014-01-01 25.280833
## 2014-01-02 39.924167
## 2014-01-03  4.992083
## 2014-01-04  4.091667
## 2014-01-05 13.587500
## 2014-01-06 47.885417

# Get back raw matrix
head(coredata(x, fmt = FALSE))
##           [,1]
## [1,] 25.280833
## [2,] 39.924167
## [3,]  4.992083
## [4,]  4.091667
## [5,] 13.587500
## [6,] 47.885417

# Extract index/time
head(index(x))
## [1] "2014-01-01" "2014-01-02" "2014-01-03" "2014-01-04" "2014-01-05"
## [6] "2014-01-06"

# Class attributes - index class
indexClass(x)
## Warning: Funkcja 'indexClass' jest przestarzała.
## Użyj w zamian 'tclass'.
## Zobacz help("Deprecated") oraz help("xts-deprecated").
## [1] "Date"
```

Following, multiple functions for solving the problem of missing values are presented.

```
#Omit NA values in x
x_ommited <- na.omit(x)

#Fill missing values in x using last observation
x_na_last <-na.locf(x)

#Fill missing values in x using next observation
x_na_next <- na.locf(x,  fromLast=TRUE)

#Interpolate NAs
x_na_interpolated <- na.approx(x)
```

Lastly, some general functions are listed.

```
#Value of data_xts in index of data_xts
head(.indexwday(x))
## [1] 3 4 5 6 0 1

#First and last observation
start(x)
## [1] "2014-01-01"
```

```
end(x)
## [1] "2018-12-31"

# Data structure of x
str(x)
## An xts object on 2014-01-01 / 2018-12-31 containing:
##   Data:    double [1826, 1]
##   Index:   Date [1826] (TZ: "UTC")
```

Data import / export

XTS objects can not only be created, but also imported and the converted. Following are listed multiple ways to import and export data, including efficient usage of zoo.

```
# Convert imported data to xts (needs to contain datetime class variable)
data$datetime <- as.Date(data$datetime)
data_xts <- as.xts(data)
head(data_xts)
##                value
## 2014-01-01 25.280833
## 2014-01-02 39.924167
## 2014-01-03  4.992083
## 2014-01-04  4.091667
## 2014-01-05 13.587500
## 2014-01-06 47.885417

# class type
class(data_xts)
## [1] "xts" "zoo"
```

```
#Read data using zoo
#as.xts(read.zoo("file_name"))

# Save to external file
write.zoo(data_xts, file = "xts_file_name.csv", sep = "'")

# Save for R use - optimized for objects like xts -> fast read
saveRDS(data_xts, "xts_RSD_file_name.rds")
head(readRDS("xts_RSD_file_name.rds"))
##                value
## 2014-01-01 25.280833
## 2014-01-02 39.924167
## 2014-01-03  4.992083
## 2014-01-04  4.091667
## 2014-01-05 13.587500
## 2014-01-06 47.885417
```

Periods and periodicity

Time series analysis is inherently related to the time space/intervals in which data was observed. Following listed are functions related to periodic nature of TS.

Estimation of frequency and time-span of observations.

```
periodicity(data_xts)
## Daily periodicity from 2014-01-01 to 2018-12-31
```

Conversion of xts to monthly, yearly OHLC.

OHLC - first (Opening) and last (Close) value from period, plus Highest and Lowest

```
data_xts_monthly <- to.monthly(data_xts)
head(data_xts_monthly)
##          data_xts.Open data_xts.High data_xts.Low data_xts.Close
## sty 2014      25.28083      56.20292    4.0916667      10.879583
## lut 2014      26.27000      45.36875    0.4779167       2.280417
## mar 2014       0.77875      51.54125    0.7787500      21.251667
## kwi 2014      33.83833      37.63333    9.9533333      28.877083
## maj 2014      22.85458      51.12042   22.8545833      38.532500
## cze 2014      49.47667      64.63208   19.7129167      49.882500
#Count months
nmonths(data_xts_monthly)
## [1] 60

data_xts_yearly <- to.yearly(data_xts)
data_xts_yearly
##            data_xts.Open data_xts.High data_xts.Low data_xts.Close
## 2014-12-31      25.28083      71.06167    0.4779167      46.702500
## 2015-12-31      54.32208      66.40833   16.3500000      27.332083
## 2016-12-31      30.72500      66.92250    5.4566667      51.090833
## 2017-12-31      60.20125      91.88333    7.6391667       7.639167
## 2018-12-31      37.69000      75.93417    4.4962500      63.454583
```

Conversion of xts to monthly periods, while aggregating values by mean.

```
data_xts_monthly_mean <- apply.monthly(data_xts, mean)
tail(data_xts_monthly_mean)
##               value
## 2018-07-31 62.08892
## 2018-08-31 64.37499
## 2018-09-30 71.28331
## 2018-10-31 65.07607
## 2018-11-30 61.86708
## 2018-12-31 61.89285
```

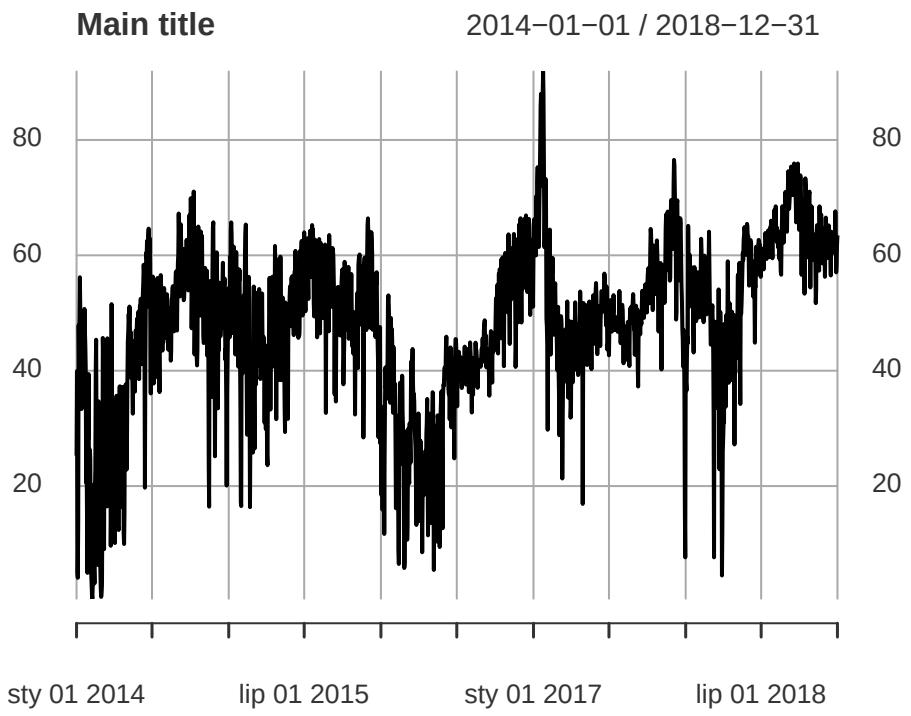
Duplicates removal

```
data_xts_unique <- make.index.unique(data_xts,drop=TRUE)
head(data_xts_unique)
##                value
## 2014-01-01 25.280833
## 2014-01-02 39.924167
## 2014-01-03  4.992083
## 2014-01-04  4.091667
## 2014-01-05 13.587500
## 2014-01-06 47.885417
```

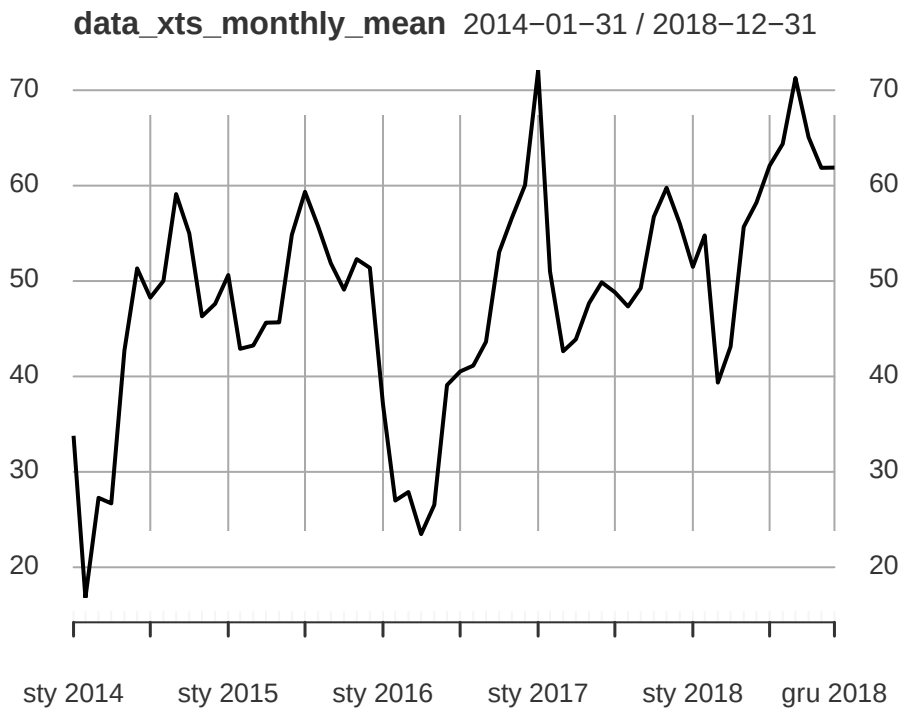
Visualization

Finally, this section presents visualization of xts. By nature, xts objects can skip xts in plot.xts, which is required for standard data frames.

```
# If plotted object is xts object, simple plot() can be used
plot.xts(data_xts, main = "Main title")
```




```
plot(data_xts_monthly_mean)
```




```
lines(data_xts_monthly_mean, col = "green", lwd = 2) #lwd - line fitnes adjustment
```