Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

RJDemetra: An R Interface To JDemetra+ Seasonal Adjustment Software

Anna Michalek
European Central Bank

Alain Quartier-la-Tente
Insee

Abstract

RJDemetra provides an interface between R and JDemetra+, the only seasonal and trading days adjustment software officially recommended by Eurostat to the members of the European Statistical System and the European System of Central Banks. **RJDemetra** offers access to main options and outputs of JDemetra+, including the two leading seasonal adjustment methods TRAMO-SEATS and X-13ARIMA. Thus, it is possible to use user-defined or pre-specified specifications and to estimate a RegARIMA model including automatic outlier and ARIMA detection, moving holiday effects and user-defined regressors. With **RJDemetra** it is also possible to read and write JDemetra+ workspaces that are used in production. Thus, thanks to all the resources available in R, it offers large possibilities to develop tools to improve the production of seasonal adjusted series.

Keywords: R, seasonal adjustment, calendar effects, ARIMA, outliers, time series.

1. Introduction

Since the 20th century, more and more infra-annual statistics are produced, especially by national institutes, to analyse the short-term evolution of economies. It is for example the case of the gross domestic product (GDP), unemployment rate, household consumption of goods and industrial production indices. However, most of those time series are affected by seasonal and trading days effects. A seasonal effect is an effect that occurs in the same calendar month with similar magnitude and direction from year to year. For instance, automobile production is usually lower during summer, due to holidays, and chocolate sales are usually higher in December, due to Christmas. Trading days effect appears when a time series is affected by calendar month's weekday composition. For example retail sales are usually higher on Saturday, thus it is likely that they will be higher in months with a surplus of weekend days.

Seasonal and trading days effects can hamper the analysis of infra-annual movements of a time series or the spatial comparison. This is the reason why time series are often seasonally and trading days adjusted, where seasonal adjustment is the process of removing the effects of seasonal and trading day fluctuations.

2. Theory behind seasonal adjustment

The most popular seasonal adjustment methods are TRAMO-SEATS[1] (Gómez and Maravall 1996; Caporello and Maravall 2004), a parametric method based on ARIMA models, and X-13ARIMA[2] (Findley, Monsell, Bell, Otto, and Chen 1998; Ladiray and Quenneville 2001), a non-parametric method based on moving averages. Both methods are recommended by Eurostat and the European Central Bank (ECB) for adjusting economic indicators. These two methods proceed in two steps, summarized in figure 1.

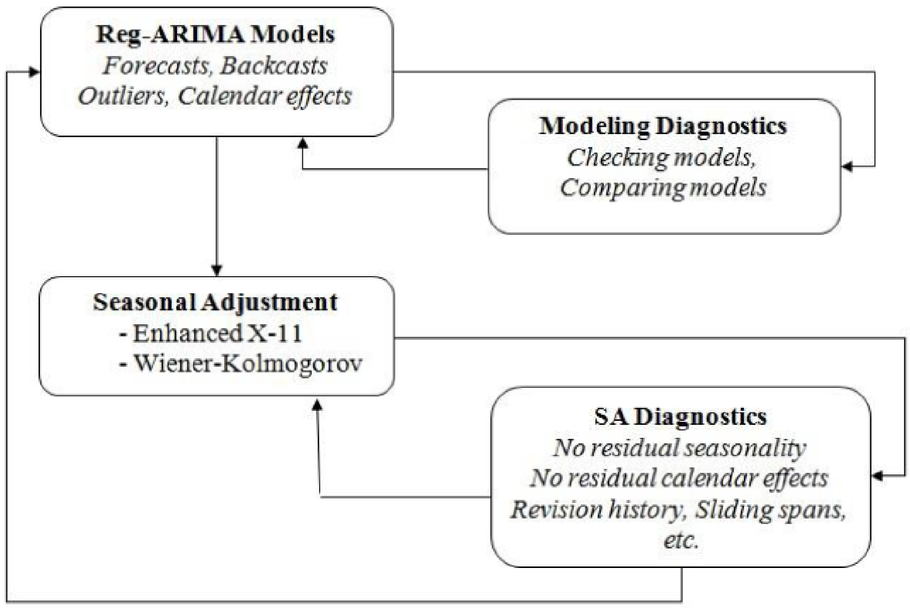


Figure 1: X-13ARIMA and TRAMO-SEATS 2-step process: pre-adjustment and decomposition.

The **first step**, called **pre-adjustment** or **linearisation**, consists of pre-adjusting the time series by removing the deterministic effects and estimating missing observations. Next, in the **second part** of seasonal adjustment, called the **decomposition**, the pre-adjusted series is decomposed in order to determine the seasonal component. As a result of this process, the final seasonally adjusted series shall be free of seasonal and calendar-related movements.

The pre-adjustment step is very similar in X-13ARIMA and in TRAMO-SEATS (section 2.1), whereas the decomposition differs between the two methods. In X-13ARIMA, the X-11 algorithm decomposes the time series by means of linear filters (section 2.2). In TRAMO-

[1] The program TRAMO-SEATS was developed by Gianluca Caporello and Agustin Maravall — with programming support from Domingo Perez and Roberto Lopez — at the Bank of Spain. It is based on the program TRAMO-SEATS, previously developed by Victor Gomez and Agustin Maravall.

[2] The program X-13ARIMA is a produced, distributed, and maintained by the US-Census Bureau.

SEATS, SEATS (Signal Extraction in ARIMA Time Series) decomposes the observed series with a ARIMA-model based method (section 2.3).

2.1. Pre-adjustment with TRAMO and RegARIMA models

As mentioned before, the **first step** of seasonal adjustment consists of pre-adjusting the time series by removing from it the deterministic effects like outliers, calendar and regression effects. This step estimates also the missing observations, as well as produces forecasts and backasts of the pre-adjusted series which allows applying linear filters at both ends of the series in the decomposition part of the seasonal adjustment. All this is achieved with a **RegARIMA** model (model with ARIMA errors) as specified below.

$$z_t = y_t\beta + x_t$$

where

- z_t - is the original series;
- $\beta = (\beta_1, \ldots, \beta_n)$ - a vector of regression coefficients;
- $y_t = (y_{1t}, \ldots, y_{nt})$ - n regression variables (outliers, calendar effects, user-defined variables);
- x_t - a disturbance that follows the general ARIMA process:
- $\phi(B)\delta(B)x_t = \theta(B)a_t$; $\phi(B), \delta(B)$ and $\theta(B)$ are the finite polynomials in B; a_t is a white-noise variable with zero mean and a constant variance.

The polynomial $\phi(B)$ is a stationary autoregressive (AR) polynomial in B, which is a product of the stationary regular AR polynomial in B and the stationary seasonal polynomial in B^s:

$$\phi(B) = \phi_p(B)\Phi_{bp}(B^s) = (1 + \phi_1 B + \cdots + \phi_p B^p)(1 + \Phi_1 B^s + \cdots + \Phi_{bp} B^{bps})$$

where:

- p - number of regular AR terms (in the package and in JDemetra+ $p \leq 3$);
- bp - number of seasonal AR terms (in the package and in JDemetra+ $bp \leq 1$);
- s - number of observations per year (frequency of the time series).

The polynomial $\theta(B)$ is an invertible moving average (MA) polynomial in B, which is a product of the invertible regular MA polynomial in B and the invertible seasonal MA polynomial in B^s:

$$\theta(B) = \theta_q(B)\Theta_{bq}(B^s) = (1 + \theta_1 B + \cdots + \theta_q B^q)(1 + \Theta_1 B^s + \cdots + \Theta_{bq} B^{bqs})$$

where:

- q - number of regular MA terms (in the package and in JDemetra+ $q \leq 3$);
- bq - number of seasonal MA terms (in the package and in JDemetra+ $bq \leq 1$);

The polynomial $\delta(B)$ is the non-stationary AR polynomial in B (unit roots):

$$\delta(B) = (1 - B)^d (1 - B^s)^{d_s}$$

where:

- d - regular differencing order (in the package and in JDemetra+ $d \leq 1$);
- d_s - seasonal differencing order (in the package and in JDemetra+ $d_s \leq 1$);

Furthermore, in this step an automatic modelling is implemented (in both methods) to: determine the decomposition of the series, detect outliers and calendar effects and to adjust residuals to an ARIMA models. A detailed description can be found in Gómez and Maravall (1998).

2.2. Decomposition with X-11

In this step, the pre-adjusted series (y) is decomposed into the following components: trend-cycle (t), seasonal component (s) and irregular component (i), where the decomposition can be:

- additive ($y = t + s + i$);
- multiplicative ($y = t \times s \times i$);
- log-additive ($\log(y) = \log(t) + \log(s) + \log(i)$);
- pseudo-additive ($y = t \times (s + i - 1)$).

In X-11, which is an iterative non-parametric method, the decomposition is achieved by means of linear filters (Findley *et al.* 1998; Ladiray and Quenneville 2001). The basic procedure consists of a simple 3-step algorithm:

1) Estimate the trend by means of moving averages;
2) Remove the trend and leave the seasonal and irregular components;
3) Estimate the seasonal component using moving averages.

At each step, the program selects a moving average among a large set of predefined smoothers, according to the characteristics of the series. X-11 also incorporates an automatic detection and correction of (additive) outliers to make the use of linear filters more robust.

2.3. Decomposition with SEATS

SEATS is a program for decomposing time series into their unobserved components following an ARIMA model that extracts from a time series its different signals (Gómez and Maravall 1996; Caporello and Maravall 2004). The decomposition can be:

- additive or ;

- multiplicative (equivalent to an additive model after taking the logarithm).

SEATS decomposes the linearized series into the following components:

- **trend-cycle component:** captures the low-frequency variation of the series and displays a spectral peak at frequency 0;
- **seasonal component:** captures the spectral peaks at seasonal frequencies;
- **irregular component:** captures erratic, white-noise behaviour, and hence has a flat spectrum;
- **transitory component:** a zero-mean stationary component that picks up transitory fluctuations that should not contaminate the trend-cycle or seasonal component and are not white-noise.

The components are determined and fully derived from the structure of the ARIMA model for the observed series.

The decomposition assumes orthogonal components, and each one will have in turn an ARIMA expression. In order to identify the components, it is required that (except for the irregular one) they are clean of noise. This is called the "canonical" property, and implies that no additive white noise can be extracted from a component that is not the irregular one. In this way, the variance of the irregular component is maximized, and the trend-cycle and seasonal component are kept as stable as possible (compatible with the stochastic nature of model).

3. JDemetra+ and RJDemetra

JDemetra+ is a tool for seasonal adjustment (SA) developed by the National Bank of Belgium (NBB) in cooperation with the Deutsche Bundesbank and Eurostat in accordance with the Guidelines of the European Statistical System (ESS) (Eurostat 2015). It implements the concepts and algorithms used in the two leading seasonal adjustment methods: TRAMO-SEATS and X-13ARIMA. Those methods have been re-engineered using an object-oriented approach that facilitates estimations handling, extensions and modifications.

JDemetra+ has been officially recommended, since 2 February 2015, to the members of the ESS and the European System of Central Banks as software for seasonal and calendar adjustment of official statistics.

Besides seasonal adjustment, JDemetra+ bundles other time series models that are useful in the production and analysis of economic statistics, including outlier detection, nowcasting, temporal disaggregation or benchmarking. More details on the methodology used in JDemetra+ can be found in the JDemetra+ manuals and user guides (Grudkowska 2015a,b).

The package **RJDemetra** (Quartier-la-Tente, Michalek, Palate, and Baeyens 2019) provides an R interface to the seasonal adjustment software JDemetra+. **RJDemetra** uses Java libraries of JDemetra+, therefore relies on the **rJava** (Urbanek 2018) package. Consequently Java SE 8 or later versions are required. The package allows to:

- perform seasonal adjustment with TRAMO-SEATS and X-13ARIMA with pre-defined (section 4) and user-defined specifications (section 6);
- access all outputs available in JDemetra+ (section 5);
- import and export JDemetra+ workspaces (section 7).

It can be installed from CRAN:

```
R> install.packages("RJDemetra")
```

The development version can be installed from GitHub with **devtools** (Wickham, Hester, and Chang 2018):

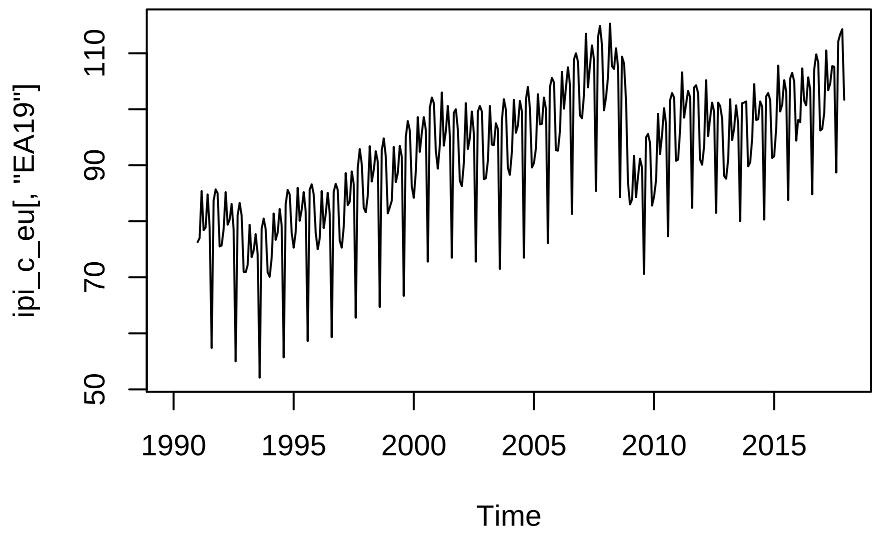
```
R> devtools::install_github("jdemetra/rjdemetra")
```

For the moment the seasonal and trading day adjustment can only be done for monthly, bimonthly (for TRAMO-SEATS only), quarterly and half-yearly data. By the end of 2020 JDemetra+, and therefore **RJDemetra**, will be extended to be compatible with every biannual frequency (daily, weekly, etc.).

3.1. Dataset

The package **RJDemetra** includes the `sts_inpr_m` database of the Eurostat, which contains monthly industrial production indices in manufacturing for the European Union. It contains 37 time series from January 1990 to December 2017 which are considered to be affected by seasonal and trading days effects. The data are `ts` objects and can be accessed using the `ipi_c_eu` object. The following snippet of code plots the industrial production index of the euro area (EA19):

```
R> library(RJDemetra)
R> plot(ipi_c_eu[, "EA19"])
```



3.2. Print styling

By default, a colour styling is used for the `print` methods of the objects created by **RJDemetra**. It can causes troubles with some outputs — for example with **rmarkdown** (Xie, Allaire, and Grolemund 2018) — and can be disabled in each `print` function with the argument `enable_print_style = FALSE` or setting the global option `enable_print_style` to `FALSE`:

```
R> options(enable_print_style = FALSE)
```

4. Estimate a pre-defined RegARIMA and seasonal adjustment model

As in JDemetra+, **RJDemetra** allows to perform seasonal adjustment using pre-defined model specifications that are the most common specifications and are recommended to users for the start of their analysis. They are separately defined for TRAMO-SEATS and X-13ARIMA methods. It is also possible to perform only the first step of seasonal adjustment; i.e. the RegARIMA estimation. The pre-defined model specifications are described in detail in tables 1 and 2. They are identical for pre-adjustment (column 1) and for seasonal adjustment (column 2). The settings described in tables 1 and 2 refer to:

- Transformation: test to choose between an additive decomposition (no transformation) and a multiplicative decomposition (logarithmic transformation).

- Pre-adjustment for leap-year (not available for TRAMO): in the case of a multiplicative decomposition a correction of the February values is applied to the original series (before

Table 1: Pre-defined specification for TRAMO and TRAMO-SEATS

Specification								
TRAMO	TRAMO-SEATS	Trans-formation	Pre-adjust-ment for leap-year	Working days	Trading days	Easter effect	Outliers	ARIMA model
TR0	RSA0	no	no	no	no	no	no	(0,1,1)(0,1,1)
TR1	RSA1	test	no	no	no	no	test	(0,1,1)(0,1,1)
TR2	RSA2	test	no	test	no	test	test	(0,1,1)(0,1,1)
TR3	RSA3	test	no	no	no	no	test	AMI
TR4	RSA4	test	no	test	no	test	test	AMI
TR5	RSA5	test	no	no	yes	test (Standard)	test	AMI
TRfull (default)	RSAfull (de-fault)	test	yes	test	test	test (Include Easter)	test	AMI

transformation). The original values in February are multiplied by $\frac{28.25}{29}$ for leap years, by $\frac{28.25}{28}$ for non-leap years and values for other months are not modified. In the case of multiplicative models, this is equivalent to adding a leap year regressor (Bell 1992).

- Working days/trading days: test for the presence of working day/trading day effects. In TRAMO an automatic choice between working days and trading days regressors is done with "RSAFull".

- Easter: pre-test for the presence of the Easter effect. For TRAMO-SEATS the default length of the Easter effect is 6 days and for X-13ARIMA an automatic detection of the duration is done (1, 8 or 15 days).

- Outliers: an automatic identification of three types of outliers: AO (additive outlier), LS (level shift) and TC (transitory change), using a default critical value. The automatic identification of SO (seasonal outlier) is not enabled by default.

- ARIMA model: the choice between fixing the ARIMA model structure to (0,1,1)(0,1,1) (Airline model) or searching for ARIMA model orders using an automatic model identification procedure. The Airline model is used as a default model in several TRAMO-SEATS and X-13ARIMA specifications as it has been shown in several studies that it is appropriate in many cases for real seasonal monthly or a quarterly time series. Moreover, the Airline model approximates well many other models and provides an excellent "benchmark" model (Maravall 2009).

To estimate a model with a pre-defined specification the following four functions can be used in **RJDemetra**:

- RegARIMA
  - X-13ARIMA method: `regarima_x13()`
  - TRAMO-SEATS method: `regarima_tramoseats()`
- Seasonal adjustment

Table 2: Pre-defined specification for RegARIMA and X-13ARIMA

Specification								
RegARIMA	X-13ARIMA	Trans-formation	Pre-adjust-ment for leap-year	Working days	Trading days	Easter effect	Outliers	ARIMA model
RG0		no	no	no	no	no	no	(0,1,1)(0,1,1)
RG1	RSA1	test	no	no	no	no	test	(0,1,1)(0,1,1)
RG2c	RSA2c	test	test	test	no	test	test	(0,1,1)(0,1,1)
RG3	RSA3	test	no	no	no	no	test	AMI
RG4c	RSA4c	test	test	test	no	test	test	AMI
RG5c (default)	RSA5 (default)	test	test	no	test	test	test	AMI

  – X-13ARIMA method: `x13()`
  – TRAMO-SEATS method: `tramoseats()`

Where the second argument refers to model specifications as described in table 1 and 2.

For example:

```
R> myseries <- ipi_c_eu[, "EA19"]
R> regx13 <- regarima_x13(myseries, spec = "RG5c")
R> regts <- regarima_tramoseats(myseries, spec = "TRfull")
R> sax13 <- x13(myseries, spec = "RSA3", userdefined = NULL)
R> sats <- tramoseats(myseries, spec = "RSAfull", userdefined = NULL)
```

As mentioned before the model specifications can be modified by users, including the possibility to incorporate user-defined regressors. How to do it is described in section 6.

5. Class object structure

To recap, section 4 presented how to run a RegARIMA and complete seasonal adjustment estimation with pre-defined model specifications. This section, in turn, presents the outcome of it.

As a result of seasonal adjustment estimation (e.g. function `x13` or `tramoseats`) a S3 class object (`sa_object`) is created. It has a class `c("SA", "X13")` or `c("SA", "TRAMO_SEATS")` depending on the used estimation method. The `sa_object` is a list of the following S3 class sub-objects: **regarima**, **decomposition**, **final**, **diagnostics** and **user_defined**. The complete structure of the `sa_object` is presented in table 3 for seasonal adjustment made with `x13` and in table 4 for seasonal adjustment made with `tramoseats`. Independently which of the two estimation methods is used, the `regarima`, `final` and `diagnostics` objects contain the same components, though with different classes (see tables 3 and 4). Whereas, the object decomposition differs for the two methods. The object user_defined is empty unless additional output was requested by the user (see sub-section 5.5). Finally, when estimating RegARIMA only the `regarima` object is created. For each of the class `print` and `plot` methods are defined. And all the plots methods are detailed in table 5.

Table 3: `SA` object structure (seasonal adjustment made with `x13`)

Object	Level	Type [RJDemetra S3 class]
sa_object	0	list [SA, X13]
regarima	**1**	**list [regarima, X13]**
specification	2	list
estimate	3	data.frame
transform	3	data.frame
regression	3	list
userdef	4	list
specification	5	data.frame
outliers	5	data.frame or NA(empty)
variables	5	list
series	6	mts, ts, matrix or NA(empty)
description	6	data.frame or NA(empty)
trading.days	4	data.frame
easter	4	data.frame
outliers	3	data.frame
arima	3	list
specification	4	data.frame
coefficients	4	data.frame or NA(empty)
forecast	3	data.frame
span	3	data.frame
arma	2	vector - numeric
arima.coefficients	2	matrix
regression.coefficients	2	matrix
loglik	2	matrix
model	2	list
spec_rslt	3	data.frame
effects	3	mts, ts, matrix
residuals	2	ts
residuals.stat	2	list
st.error	3	numeric
tests	3	data.frame [regarima_rtests]
forecast	2	mts, ts, matrix
decomposition	**1**	**list [decomposition_X11]**
specification	2	data.frame [X11_spec]
mode	2	character
mstats	2	matrix
si_ratio	2	mts, ts, matrix
s_filter	2	vector - character
t_filter	2	character

final	**1**	**list [finals]**
series	2	mts, ts, matrix
forecasts	2	mts, ts, matrix
diagnostics	**1**	**list [diagnostics]**
variance_decomposition	2	data.frame
combined_test	2	list [combined_test]
tests_for_stable_seasonality	3	data.frame
combined_seasonality_test	3	character
residuals_test	2	data.frame
user_defined	**1**	**list [user_defined]**

Table 4: `SA` object structure (seasonal adjustment made with `tramoseats`)

Object	Level	Type [RJDemetra S3 class]
sa_object	0	list [SA, X13]
regarima	**1**	**list [regarima, X13]**
specification	2	list
estimate	3	data.frame
transform	3	data.frame
regression	3	list
userdef	4	list
specification	5	data.frame
outliers	5	data.frame or NA(empty)
variables	5	list
series	6	mts, ts, matrix or NA(empty)
description	6	data.frame or NA(empty)
trading.days	4	data.frame
easter	4	data.frame
outliers	3	data.frame
arima	3	list
specification	4	data.frame
coefficients	4	data.frame or NA(empty)
forecast	3	data.frame
span	3	data.frame
arma	2	vector - numeric
arima.coefficients	2	matrix
regression.coefficients	2	matrix
loglik	2	matrix
model	2	list
spec_rslt	3	data.frame
effects	3	mts, ts, matrix
residuals	2	ts

residuals.stat	2	list
st.error	3	numeric
tests	3	data.frame [regarima_rtests]
forecast	2	mts, ts, matrix
decomposition	**1**	**list [decomposition_seats]**
specification	2	data.frame [seats_spec]
mode	2	character
model	2	list
model	3	matrix or empty list
sa	3	matrix or empty list
trend	3	matrix or empty list
seasonal	3	matrix or empty list
transitory	3	matrix or empty list
irregular	3	matrix or empty list
linearized	2	mts, ts, matrix
components	2	mts, ts, matrix
final	**1**	**list [finals]**
series	2	mts, ts, matrix
forecasts	2	mts, ts, matrix
diagnostics	**1**	**list [diagnostics]**
variance_decomposition	2	data.frame
combined_test	**2**	**list [combined_test]**
tests_for_stable_seasonality	3	data.frame
combined_seasonality_test	3	character
residuals_test	2	data.frame
user_defined	**1**	**list [user_defined]**

Table 5: Plots available with the **RJDemetra** package.

Object class (`x` object)	Method	Description
`regarima`	`plot(x, which = 1)`	Plot of residuals
`regarima`	`plot(x, which = 2)`	Histogram of standardized residuals and density
`regarima`	`plot(x, which = 3)`	Normal quantile-quantile (Q-Q) plot of standardized residuals
`regarima`	`plot(x, which = 4)`	Autocorrelation function (ACF) of residuals
`regarima`	`plot(x, which = 5)`	Partial autocorrelation function (PACF) of residuals
`regarima`	`plot(x, which = 6)`	Raw and linearized series
`regarima`	`plot(x, which = 7)`	Plots 3 graphics: linearized series, calendar effects and outliers effects

`decomposition_X11`, `decomposition_SEATS`	`plot(x)`	S-I ratio: seasonal-irregular (S-I) component and the seasonal factors for each period of the time series (months or quarters)
`final`	`plot(x, type_chart = sa-trend)`	Plots the raw series, the seasonal adjusted series and the trend
`final` or `SA`	`plot(x, type_chart = cal-seas-irr)`	Plots the calendar effects, the seasonal component and the irregular

5.1. RegARIMA

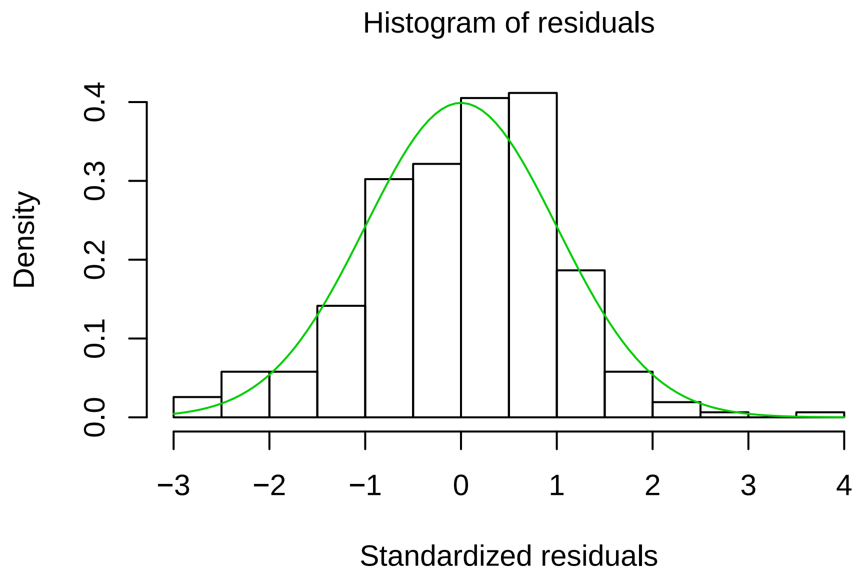
The `regarima` object contains provided by users model specification (`specification`; level 2 of the `sa_object`), the estimated coefficients for the ARIMA processes (`arima.coefficients`) and for the regressors (`regression.coefficients`), including ARMA orders (`arma`). The object includes also model quality measures (`loglik`), RegARIMA specification after its estimation with the estimated effects (e.g. linearized input series or outliers)(`model`), residuals of the RegARIMA model (`residuals`), several tests' results for the residuals (`residuals.stat`) and finally the forecast of the pre-adjusted series (`forecasts`). All this information can be extracted individually by users by referring to different parts of the S3 class object or a predefined output can be used with functions `print()` or `summary()`. Furthermore graphical presentations are also available with the function `plot()` that displays a set of graphs. For `regarima` by default the first six graphs are displayed, but specific ones can be chosen within the argument `which`. Table 5 summarises all the graphs available for the `sa_object`, as well as its `plot()` options.

```
R> sax13$regarima

y = regression model + arima (1, 1, 2, 0, 1, 1)
Log-transformation: no
Coefficients:
          Estimate Std. Error
Phi(1)     -0.7603      0.096
Theta(1)   -1.1757      0.095
Theta(2)    0.4551      0.053
BTheta(1)  -0.5433      0.049

             Estimate Std. Error
AO (1-2016)     4.291      0.883
LS (1-2009)    -6.210      0.947
LS (11-2008)   -5.806      0.948
TC (3-2009)    -3.967      0.908


Residual standard error: 1.187 on 311 degrees of freedom
Log likelihood = -496.8, aic =  1012 aicc =  1012, bic(corrected for length) = 0.4898
```

```
R> summary(sax13$regarima)
```

```
y = regression model + arima (1, 1, 2, 0, 1, 1)

Model: RegARIMA - X13
Estimation span: from 1-1991 to 12-2017
Log-transformation: no
Regression model: no mean, no trading days effect, no leap year effect, no Easter effect,

Coefficients:
ARIMA:
            Estimate Std. Error  T-stat Pr(>|t|)
Phi(1)      -0.76028    0.09609  -7.912 4.49e-14 ***
Theta(1)    -1.17573    0.09515 -12.356  < 2e-16 ***
Theta(2)     0.45508    0.05285   8.612 4.44e-16 ***
BTheta(1)   -0.54327    0.04932 -11.015  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Regression model:
             Estimate Std. Error T-stat Pr(>|t|)
AO (1-2016)    4.2913     0.8832  4.859 1.88e-06 ***
LS (1-2009)   -6.2105     0.9469 -6.559 2.26e-10 ***
LS (11-2008)  -5.8056     0.9479 -6.125 2.74e-09 ***
TC (3-2009)   -3.9673     0.9078 -4.370 1.69e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


Residual standard error: 1.187 on 9 degrees of freedom
Log likelihood = -496.8, aic =  1012, aicc =  1012, bic(corrected for length) = 0.4898
```

```
R> plot(sax13$regarima, which = 2)
```



5.2. Decomposition

As briefly discussed above, the decomposition method differs between TRAMO-SEATS and X-13ARIMA, where SEATS is based on ARIMA-model and X-11-algorithm on linear filters. Consequently the composition of this object differs between the two methods (tables 3 and 4). The only common part is the first two sub-objects with the model specification (`specification`) and information on the decomposition mode (`mode`; e.g.: additive).

Then, the `decomposition_X11` object comprises quality measures on the decomposition (`mstats`), namely the M and Q statistics. It contains also the final unmodified S-I ratios $d8$ and final seasonal factors $d10$ (`si_ratio`), as well as the information on the final seasonal filter (`s_filter`) and trend filter(`t_filter`). The code below presents the output for X-11 decomposition:

```
R> sax13$decomposition

 Monitoring and Quality Assessment Statistics:
      M stats
M(1)    0.028
M(2)    0.033
M(3)    0.338
M(4)    0.376
M(5)    0.366
M(6)    0.067
M(7)    0.066
M(8)    0.157
M(9)    0.073
```

```
M(10)   0.145
M(11)   0.120
Q       0.156
Q-M2    0.171

Final filters:
Seasonal filter:  3x5
Trend filter:  13 terms Henderson moving average
```

As a reminder, in SEATS it is assumed that each component of the linearized series (received from TRAMO) is an outcome of a linear stochastic process and SEATS estimates an ARIMA model for each component (i.e. trend, seasonal, transitory and irregular). Therefore the `decomposition_SEATS` object contains the information on the estimated ARIMA models (`model`), the linearized components - as obtained from TRAMO (`linearized`) -, and the theoretical components calculated from the ARIMA models (`components`). The code below presents the output for the SEATS decomposition, with the information on the ARIMA models:

```
R> sats$decomposition

Model
AR :  1 + 0.094056 B - 0.158875 B^2 - 0.294600 B^3
D :  1 - B - B^12 + B^13
MA :  1 - 0.510600 B^12


SA
AR :  1 + 0.094056 B - 0.158875 B^2 - 0.294600 B^3
D :  1 - 2.000000 B + B^2
MA :  1 - 0.937923 B - 0.015753 B^2 - 0.007005 B^3 + 0.015440 B^4 - 0.001104 B^5
Innovation variance:  0.5777661

Trend
AR :  1 - 0.711394 B
D :  1 - 2.000000 B + B^2
MA :  1 - 0.312018 B - 0.965498 B^2 + 0.346519 B^3
Innovation variance:  0.07090723

Seasonal
D :  1 + B + B^2 + B^3 + B^4 + B^5 + B^6 + B^7 + B^8 + B^9 + B^10 + B^11
MA :  1 + 1.314878 B + 1.722951 B^2 + 2.227262 B^3 + 2.229462 B^4 + 2.119339 B^5 + 1.92645
Innovation variance:  0.08441922

Transitory
AR :  1 + 0.805449 B + 0.414116 B^2
MA :  1 - 0.452913 B - 0.547087 B^2
Innovation variance:  0.02525112
```

```
Irregular
Innovation variance:  0.07006354
```

5.3. Final

The final object has a simple structure as it includes the input series, final seasonally adjusted series and the final components (i.e. t - trend-cycle, s - seasonal component and i - irregular component) (`series`), as well as their forecasts (`forecasts`).

```
R> sats$final

Last observed values
             y       sa        t             s           i
Jan 2017  96.5 102.9317 102.6366  -6.431739755  0.29515069
Feb 2017  99.3 102.3815 102.7523  -3.081452963 -0.37082579
Mar 2017 110.5 103.2230 103.0587   7.276971010  0.16431902
Apr 2017 103.4 103.3950 103.4579   0.004977088 -0.06292068
May 2017 104.6 104.1023 103.8400   0.497657700  0.26230153
Jun 2017 107.7 103.8403 104.2863   3.859652578 -0.44596785
Jul 2017 107.6 105.1862 104.9032   2.413785110  0.28301107
Aug 2017  88.7 105.5484 105.5999 -16.848375561 -0.05151353
Sep 2017 112.1 106.2889 106.3305   5.811052841 -0.04159101
Oct 2017 113.4 106.9003 107.1101   6.499696169 -0.20982123
Nov 2017 114.3 108.3110 107.7487   5.988950554  0.56232415
Dec 2017 101.7 107.7534 108.1104  -6.053350690 -0.35709913

Forecasts:
               y_f     sa_f      t_f         s_f           i_f
Jan 2018 102.49061 108.4010 108.4016  -5.9103802 -0.0005820963
Feb 2018 105.85284 108.8666 108.7446  -3.0137577  0.1219532717
Mar 2018 116.21918 108.9840 109.0820   7.2351710 -0.0979861094
Apr 2018 108.90762 109.4437 109.4153  -0.5360850  0.0284199810
May 2018 110.11320 109.7634 109.7457   0.3498301  0.0176868066
Jun 2018 114.47710 110.0480 110.0740   4.4290998 -0.0260150048
Jul 2018 112.47723 110.4145 110.4009   2.0627149  0.0136293681
Aug 2018  93.98244 110.7265 110.7267 -16.7440681 -0.0002045225
Sep 2018 116.57469 111.0463 111.0518   5.5283721 -0.0054794124
Oct 2018 118.30580 111.3809 111.3764   6.9249450  0.0044980848
Nov 2018 117.56621 111.6992 111.7005   5.8670241 -0.0013538638
Dec 2018 105.59658 112.0237 112.0245  -6.4271099 -0.0007722619
```

5.4. Diagnostics

This part of the `sa_object` includes several diagnostics on the presence of seasonality in the input series and on the quality of the seasonal adjustment.

The tests for the seasonality presence (`combined_test`) are performed both on the entire series and in the last 3 years.

As regards the seasonal adjustment quality checks, they are grouped into two sets. The first looks at the contribution of each estimated component to the variance of the original series (`variance_decomposition`). The second verifies, with different tests, that there is no seasonal pattern left in the seasonally adjusted series and in the irregular component (`residuals_test`).

All the above checks (except `combined_seasonality_test`), together with a detailed description, are displayed when printing the diagnostics object.

```
R> sats$diagnostics

 Relative contribution of the components to the stationary
 portion of the variance in the original series,
 after the removal of the long term trend
 Trend computed by Hodrick-Prescott filter (cycle length = 8.0 years)
           Component
 Cycle        15.148
 Seasonal     83.993
 Irregular     0.174
 TD & Hol.     0.076
 Others        0.049
 Total        99.441

 Combined test in the entire series
 Non parametric tests for stable seasonality
                                                              P.value
   Kruskall-Wallis test                                          0.000
   Test for the presence of seasonality assuming stability      0.000
   Evolutive seasonality test                                    0.027

 Identifiable seasonality present

 Residual seasonality tests
                                      P.value
 qs test on sa                          1.000
 qs test on i                           1.000
 f-test on sa (seasonal dummies)        1.000
 f-test on i (seasonal dummies)         1.000
 Residual seasonality (entire series)   1.000
 Residual seasonality (last 3 years)    0.999
 f-test on sa (td)                      0.994
 f-test on i (td)                       0.922
```

5.5. User-defined

As presented in the tables 3 and 4 and in the previous sections the `sa_object` has a defined structure with a defined content. Nevertheless users can also extract additional output from the seasonal adjustment estimation and this will be stored under `user_defined` object, in a form of a list. In order to receive the additional output extra variables need to be defined as characters under the argument `userdefined` of the functions `x13()` or `tramoseats()`.

For example, to receive additionally tables $c10$ and $d16$ the following need to be specified in the function argument:

```
R> sa_usrdef <- x13(myseries, spec = "RSA3",
R+                  userdefined = c("decomposition.c10", "decomposition.d16"))
R> sa_usrdef$user_defined
```

```
Names of additional variables (2):
decomposition.c10, decomposition.d16
```

The list of all available variables can be obtained with the following functions:

- `user_defined_variables("X13-ARIMA")`
- `user_defined_variables("TRAMO-SEATS")`

6. Model specification: creation and modification

Users can also create their own specifications by modifying pre-defined specifications (as described in tables 1 and 2) or previously defined specifications or models. For that, there are two functions for each method (X-13ARIMA and TRAMO-SEATS) - one for the RegARIMA model and one for the entire seasonal adjustment:

- RegARIMA model: `regarima_spec_x13()` for X-13ARIMA and `regarima_spec_tramoseats()` for TRAMO-SEATS;
- seasonal adjustment: `x13_spec()` for X-13ARIMA and `tramoseats_spec()` for TRAMO-SEATS.

As mentioned above, the input of the functions can be a pre-defined JDemetra+ model specification, previously modified specification or a model.

Once the specification is created, the estimations can be performed for:

- RegARIMA model by `regarima()` and;
- seasonal adjustment with X-13ARIMA by `x13()` and with TRAMO-SEATS by `tramoseats()`.

The example below illustrates how to create its own RegARIMA model for the TRAMO-SEATS method by adding an additive outlier in October 2009:

```
R> regarima_ts_spec <- regarima_spec_tramoseats(spec = "TRfull",
R+              usrdef.outliersEnabled = TRUE,
R+              usrdef.outliersType = "AO",
R+              usrdef.outliersDate = "2009-10-01")
R> regarima_ts_model <- regarima(series = ipi_c_eu[, "EA19"],
R+                               spec = regarima_ts_spec)
R> regarima_ts_model

y = regression model + arima (3, 1, 0, 0, 1, 1)
Log-transformation: no
Coefficients:
          Estimate Std. Error
Phi(1)     0.09633      0.055
Phi(2)    -0.16551      0.055
Phi(3)    -0.29422      0.056
BTheta(1) -0.50637      0.051

             Estimate Std. Error
Monday       -0.23323      0.094
Tuesday      -0.01617      0.094
Wednesday     0.29430      0.095
Thursday     -0.35287      0.095
Friday        0.13248      0.094
Saturday      0.30763      0.095
AO (10-2009) -0.80480      0.787
AO (1-2016)   3.25565      0.807


Residual standard error: 1.226 on 311 degrees of freedom
Log likelihood = -506.6, aic =  1039 aicc =  1040, bic(corrected for length) = 0.6292
```

And how to modify the specification of the X-13ARIMA object `sa_usrdef` (defined in section 5.5) by changing the seasonal filter and performing a working day adjustment:

```
R> sa_x13_spec <- x13_spec(spec = sa_usrdef,
R+                         tradingdays.option = "WorkingDays",
R+                         x11.seasonalma = "S3X3")
R> sa_x13_model <- x13(series = ipi_c_eu[, "EA19"],
R+                     spec = sa_x13_spec)
```

Almost all the specification variables available in JDemetra+ can be used in **RJDemetra**. For more details see the help page for the corresponding function or the documentation of JDemetra+.

To prevent from wrong user specification, there are automatic checks in **RJDemetra**, like in JDemetra+. For example, to pre-specify an outlier or a user-defined variable you have to enable them (setting the parameter `usrdef.outliersEnabled` or `usrdef.varEnabled` to

TRUE); or to fix the coefficient of an outlier or a user-defined regressor you have to specify the transformation function (transform.function, it cannot be automatic). Those checks are done each time a new specification is created. Therefore, some specifications cannot be set in two stages. For example, fixing the coefficient of an outlier has to be done at the same time when the outliers are defined. The following code doesn't fix the coefficient of the outlier previously defined for January 2001:

```
R> regarima_wrong_spec <- regarima_spec_tramoseats(spec = regarima_ts_model,
R+              transform.function = "Log",
R+              usrdef.outliersCoef =  -0.8)
```

To fix it you have to re-define the outlier:

```
R> regarima_good_spec <- regarima_spec_tramoseats(spec = regarima_ts_model,
R+              transform.function = "Log",
R+              usrdef.outliersType = "AO",
R+              usrdef.outliersDate = "2009-10-01",
R+              usrdef.outliersCoef =  -0.8)
```

The documentation for the functions used to modify specifications provide information on the interdependencies between different arguments. The package also offers functions to display different parts of the model specification. These are presented under the entry specification of the package documentation. For instance, from the example above, we can check which user defined variables were enabled and with which parameters.

In the first case (wrongly specified), an outlier was pre-defined but its coefficient was not fixed:

```
R> s_usrdef(regarima_wrong_spec)

 outlier outlier.coef variables variables.coef
    TRUE        FALSE     FALSE          FALSE

R> s_preOut(regarima_wrong_spec)

  type       date coeff
1   AO 2009-10-01    NA
```

In the second case, the coefficient was correctly fixed:

```
R> s_usrdef(regarima_good_spec)

 outlier outlier.coef variables variables.coef
    TRUE         TRUE     FALSE          FALSE

R> s_preOut(regarima_good_spec)
```

```
  type       date coeff
1   AO 2009-10-01  -0.8
```

7. Manipulate JDemetra+ workspaces

RJDemetra allows to interact with JDemetra+ workspaces that can be opened by the software. A workspace includes:

- An XML file that enables users to import a workspace to JDemetra+ and to display its content;
- A folder containing several sub-folders that correspond to different types of items created by the user.

Each workspace can contain several multi-processing and each multi-processing stores results of the seasonal adjustment procedure performed with the TRAMO-SEATS or X-13ARIMA methods.

Exporting models to workspace allows to store easily the seasonal adjustment models, to change specifications with the JDemetra+ graphical interface and to give models to users unfamiliar with R.

7.1. Export a workspace

Four functions can to be used to export models:

- `new_workspace()` to create a workspace;
- `new_multiprocessing()` to create a multi-processing in a workspace;
- `add_sa_item()` to add a seasonal adjustment model to a multi-processing;
- `save_workspace()` to export the workspace.

The following commands export seasonal adjustment models computed by TRAMO-SEATS and X-13ARIMA:

```
R> myseries <- ipi_c_eu[, "EA19"]
R> sa_x13 <- x13(myseries)
R> sa_ts <- tramoseats(myseries)
```

First, to create a workspace and a multi-processing named "MP-1" the following need to be executed:

```
R> wk <- new_workspace()
R> new_multiprocessing(wk, name = "MP-1")
```

Then, the two models will be added in the multiprocessing "MP1": the name of the seasonal adjustment model computed with X-13ARIMA will be "SA with X13" and the one with TRAMO-SEATS will be "SA with TramoSeats":

```
R> add_sa_item(wk, multiprocessing = "MP-1",
R+             sa_obj = sa_x13, name =  "SA with X13")
R> add_sa_item(wk, multiprocessing =  "MP-1",
R+             sa_obj = sa_ts, name = "SA with TramoSeats")
```

The exported workspace is named "workspace.xml":

```
R> dir <- tempdir()
R> save_workspace(wk, file =  file.path(dir, "workspace.xml"))
```

7.2. Import a workspace

The following functions can be used to import a workspace:

- `load_workspace()` to load a workspace;
- `compute()` to compute multi-processing: by default a workspace contains only definitions, therefore computation is needed to get the seasonal adjustment model;
- `get_model()` to get the seasonal adjustment models;
- `get_ts()` to get the input raw time series, `get_object()` and `get_all_objects` to navigate inside a workspace (extract a multi-processing or a seasonal adjustment model), `get_name()` to get names of the multiprocessing or the seasonal adjustment models, and `count()` to count the number of multiprocessing or seasonal adjustment models.

For instance, the following need to be run to import the workspace created in section 7.1 and to get the first multiprocessing and the first seasonal adjustment model:

```
R> wk <- load_workspace(file =  file.path(dir, "workspace.xml"))
R> mp1 <- get_object(wk, 1)
R> sa_item1 <- get_object(mp1, 1)
```

To get the number of seasonal adjustment models in the multiprocessing:

```
R> count(mp1)

[1] 2
```

And to receive the name of the first seasonal adjustment model in JDemetra+:

```
R> get_name(sa_item1)
```

```
[1] "SA with X13"
```

Finally, raw time series and seasonal adjustment model can now be imported:

```
R> raw_ts <- get_ts(sa_item1)
R> compute(wk)
R> sa_model1 <- get_model(sa_item1, workspace = wk)
```

`get_ts()` and `get_model()` can also be used directly on a workspace or on a multiprocessing to import all the raw time series or all the seasonal adjustment model:

- for a multiprocessing the result is a list which each element contains the information on the seasonal adjustment model;
- for a workspace the result is a list of number of multi-processing length and which each element contains a list with the information on each seasonal adjustment model.

For example to get all raw time series of the workspace and all seasonal adjustment models of the first multi-processing the following need to be run:

```
R> all_raw_ts <- get_ts(wk)
R> sa_models_of_mp1 <- get_model(mp1, workspace = wk)
```

The imports of seasonal adjustment models from a workspace work well when they have been created through **RJDemetra**. Troubles might occur when importing a workspace created with JDemetra+, in particular:

- Seasonal adjustment models with ramp effect or intervention variables will be partially imported: the result of the imported model will be correct but changing the specification (through `x13_spec()` or `tramoseats_spec()`) will erase them.
- Seasonal adjustment models with no pre-processing (X-11 specification) are not supported: `NULL` object will be returned.

8. Advanced usage and examples

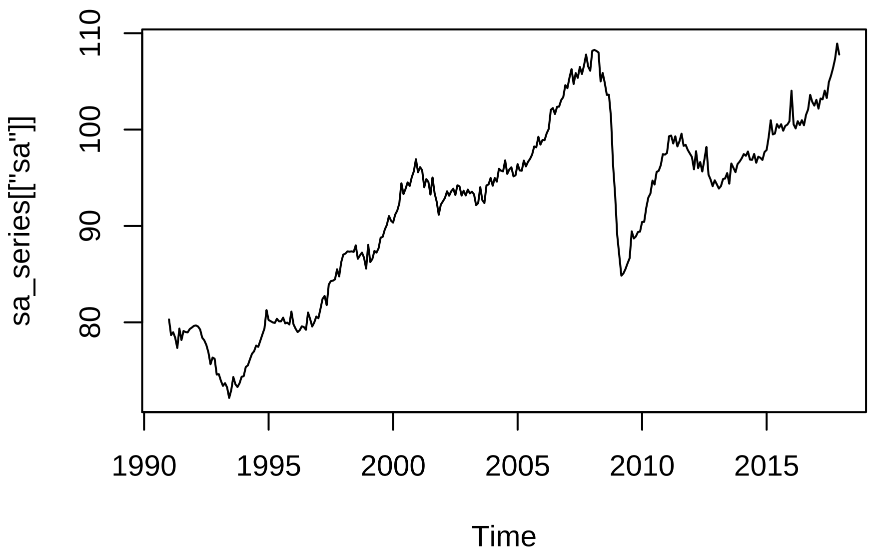
By default, `x13()`, `tramoseats()` and `regarima()` export a large number of diagnostics and indicators. This might be time-consuming, especially when dealing with many series and only a few indicators are needed. To customise the output and receive only the needed indicators, four functions extracting the associated seasonal adjustment Java model can be used: `jx13()`, `jtramoseats()`, `jregarima_x13()` and `jregarima_tramoseats()`. Three other functions can be used to manipulate these objects:

- `get_dictionary()` to get the list of indicators that can be extracted;

- `get_indicators()` to get selected indicators;
- `jSA2R()` to get the corresponding formatted seasonal adjustment or RegARIMA model.

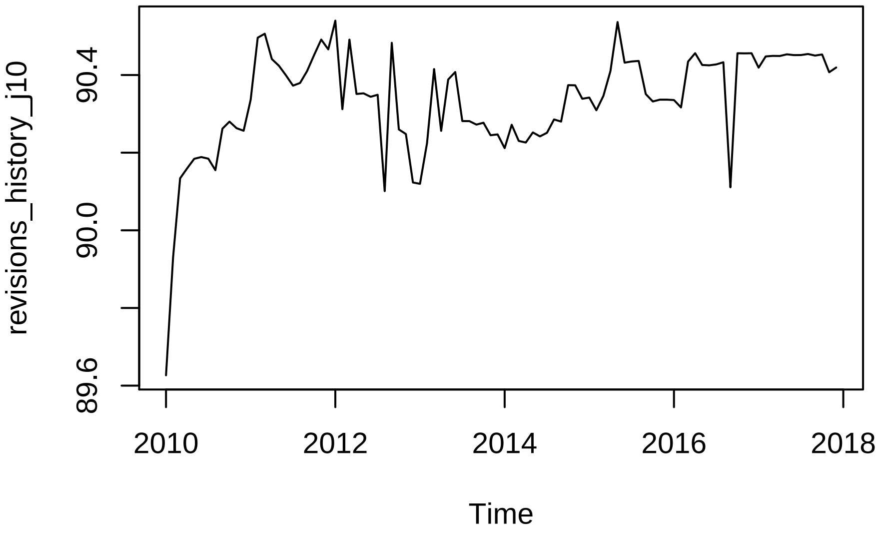
For example to only receive the seasonally adjusted time series:

```
R> sa_jx13 <- jx13(myseries)
R> head(get_dictionary(sa_jx13))

[1] "y"    "y_f"  "t"    "t_f"  "sa"   "sa_f"

R> sa_series <- get_indicators(sa_jx13, "sa")
R> plot(sa_series[["sa"]])
```



To compute the revisions of the seasonal adjusted series of January 2010 with an automatic modelling:

```
R> dates <- window(time(myseries), start = 2010)
R> revisions_history_j10 <- sapply(dates,function(last_date){
R+   sa_jx13 <- jx13(window(myseries, end = last_date))
R+   window(get_indicators(sa_jx13, "sa")[["sa"]],
R+          start = 2010,
R+          end = 2010)
R+ })
R> revisions_history_j10 <- ts(revisions_history_j10,
R+                             start = 2010, frequency = 12)
R> plot(revisions_history_j10)
```



When performing seasonal adjustment on a large database, the most common error results from the preliminary check (used to verify the quality of the input series) when excluding highly problematic series (with too many identical observations and/or missing values). In this case, the `jx13()` function will not give an error but `get_indicators()` will only return a `NULL` objects:

```
R> identical_ts <- ts(0,start = 2010, end = 2015, frequency = 12)
R> get_indicators(jx13(identical_ts), "sa", "sa_f")
```

```
$sa
NULL

$sa_f
NULL
```

To disable this preliminary check, you need to create a new specification with the parameter `preliminary.check = FALSE`:

```
R> myspec <- x13_spec(preliminary.check = FALSE)
R> my_indicators <- get_indicators(jx13(identical_ts, myspec)
R+                                 , "sa", "sa_f")
```

This might be useful when performing large scale seasonal adjustment. For example with our database on industrial production indices:

```
R> ipi_sa_series <- lapply(colnames(ipi_c_eu),function(series){
R+   sa_jx13 <- jx13(ipi_c_eu[,series], myspec)
R+   get_indicators(sa_jx13, "sa")[["sa"]]
R+ })
R> ipi_sa_series <- do.call(ts.union,ipi_sa_series)
R> colnames(ipi_sa_series) <- colnames(ipi_c_eu)
```

9. Conclusion

JDemetra+ is a powerful tool for seasonal and trading days adjustments. It implements the two leading methods, TRAMO-SEATS and X-13ARIMA, with a rich graphical interface. Besides seasonal adjustment, JDemetra+ bundles other time series models that are useful in the production or analysis of economic statistics, including outlier detection, nowcasting, temporal disaggregation or benchmarking. It is also the only software officially recommended by Eurostat for seasonal and calendar adjustment.

The package **RJDemetra** is based on libraries of JDemetra+, and therefore uses certified and tested algorithms. It also offers many opportunities to producers and researchers. JDemetra+ being widely used in production allows to:

- Easily create tools that can be used in production. For example **rjdqa** (Quartier-la-Tente 2019b) reproduces Statistics Canada dashboard, used to provide a snapshot of a single seasonal adjustment model at a point in time and to highlight some possible problems.

- Customise the output to producers' needs from a specific institution. For example the goal of **persephone** (Meraner and de Cillia 2019) is to enable an easy processing during production of seasonally adjusted series in Statistics Austria.

- Extend the current R package to use the same seasonal adjustment methods used in production. For example **ggdemetra** (Quartier-la-Tente 2019a) extend the **ggplot2** (Wickham 2016) to add seasonal adjustment statistics to plots (diagnostics, outliers, ARIMA models, seasonally adjusted series, etc.).

Moreover, **RJDemetra** manipulates directly Java objects which makes it much more efficient compared to other R packages. This is particularly important when dealing with large scale seasonal adjustment or when conducting extended studies.

The package **RJDemetra** will also evolve with JDemetra+ and integrate the new developments on seasonal adjustment methods, for example the future extension to all series frequencies (weekly, daily, etc.).

References

Bell WR (1992). "Alternative Approaches to Length of Month Adjustment." *Statistical Research Division. US Bureau of the Census Statistical Research Division Report Number: Census/SRD/RR*, **92**(01). URL `https://www.census.gov/srd/papers/pdf/rr92-17.pd`.

Caporello G, Maravall A (2004). "Program TSW: Revised reference manual." *Technical Report, Research Department*, Banco de España. URL https://www.bde.es/f/webbde/SES/Secciones/Publicaciones/PublicacionesSeriadas/DocumentosOcasionales/04/Fic/do0408e.pdf.

Eurostat (2015). "ESS Guidelines on Seasonal Adjustment." *Technical report*, Eurostat Methodologies and Working Papers, European Commission. doi:10.2785/317290. URL http://ec.europa.eu/eurostat/web/products-manuals-and-guidelines/-/KS-GQ-15-001.

Findley DF, Monsell BC, Bell WR, Otto MC, Chen BC (1998). "New Capabilities and Methods of the X-12-ARIMA Seasonal-Adjustment Program." *Journal of Business & Economic Statistics*, **16**(2), 127–152. ISSN 07350015. URL http://www.jstor.org/stable/1392565.

Gómez V, Maravall A (1996). "Programs TRAMO and SEATS. Instructions for the User." *Working papers 9628*, Banco de España.

Gómez V, Maravall A (1998). "Automatic Modeling Methods for Univariate Series." *Working papers 9608*, Banco de España. URL https://EconPapers.repec.org/RePEc:bde:wpaper:9808.

Grudkowska S (2015a). "JDemetra+ reference manual." *Department of Statistics, Narodowy Bank Polski, Warsaw.* URL https://ec.europa.eu/eurostat/cros/system/files/jdemetra_reference_manual_version_2.1_0.pdf.

Grudkowska S (2015b). "JDemetra+ user guide." *Department of Statistics, Narodowy Bank Polski, Warsaw.* URL https://ec.europa.eu/eurostat/cros/system/files/jdemetra_user_guide_version_2.2.pdf.

Ladiray D, Quenneville B (2001). *Seasonal adjustment with the X-11 Method*. Springer-Verlag Inc, Berlin; New York.

Maravall A (2009). "Identification of Reg-ARIMA Models and of Problematic Series in Large Scale Applications Program TSW (TRAMO-SEATS for Windows)." URL https://www.bde.es/f/webbde/SES/servicio/Programas_estadisticos_y_econometricos/Notas_introductorias_TRAMO_SEATS/ficheros/Large_Scale_TSW.pdf.

Meraner A, de Cillia G (2019). ***persephone**: Object-Oriented Wrappers for RJDemetra*. R package version 0.1.0, URL https://github.com/statistikat/persephone.

Quartier-la-Tente A (2019a). ***ggdemetra**: **ggplot2** Extension for Seasonal and Trading Day Adjustment with **RJDemetra***. R package version 0.2.0, URL https://CRAN.R-project.org/package=ggdemetra.

Quartier-la-Tente A (2019b). ***rjdqa**: Quality Assessment for Seasonal Adjustment*. R package version 0.1.0, URL https://CRAN.R-project.org/package=rjdqa.

Quartier-la-Tente A, Michalek A, Palate J, Baeyens R (2019). ***RJDemetra**: Interface to JDemetra+ Seasonal Adjustment Software*. R package version 0.1.3, URL https://CRAN.R-project.org/package=RJDemetra.

Urbanek S (2018). ***rJava**: Low-Level R to Java Interface*. R package version 0.9-10, URL https://CRAN.R-project.org/package=rJava.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org.

Wickham H, Hester J, Chang W (2018). ***devtools**: Tools to Make Developing R Packages Easier*. R package version 2.0.1, URL https://CRAN.R-project.org/package=devtools.

Xie Y, Allaire J, Grolemund G (2018). *R Markdown: The Definitive Guide*. Chapman and Hall/CRC, Boca Raton, Florida. ISBN 9781138359338, URL https://bookdown.org/yihui/rmarkdown.

Affiliation:

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd