

R Tools for JDemetra+ Seasonal adjustment made easier

**Anna SMYK
Alice TCHANG
(Insee)**

M 2021/01

**R Tools for JDemetra+
Seasonal adjustment made easier**

**Anna SMYK
Alice TCHANG
Insee**

Janvier 2021

Les auteurs remercient Alain Quartier-la-Tente pour son aide.

Direction de la méthodologie et de la coordination statistique et internationale
Département des Méthodes Statistiques -Timbre L001 -
88 Avenue Verdier - CS 70058 - 92541 Montrouge Cedex - France -
Tél. : 33 (1) 87 69 55 00 - E-mail : -DG75-L001@insee.fr - Site Web Insee : <http://www.insee.fr>

*Ces documents de travail ne reflètent pas la position de l'Insee et n'engagent que leurs auteurs.
Working papers do not reflect the position of INSEE but only their author's views.*

Abstract: The RJDemetra package is an interface between R and JDemetra+, the Eurostat recommended seasonal adjustment software. It offers full access to the JD+ algorithms. Several add-in packages expand its features: rjdworkspace, rjdqa, rjdmarkdown and ggdemetra. This paper aims at describing and illustrating the functionalities of those recently developed packages, when used stand-alone or in conjunction with the JDemetra+ graphical interface. It also presents an older staple tool for production: the cruncher package jwsacruncher. Eventually, we show how to take advantage of the complementary assets of these tools, according to the user's needs, be they mass production or detailed analysis.

Keywords: Time Series, R, Seasonal Adjustment, JDemetra+, RJDemetra, Quality Assessment

JEL Classification : C01, C22

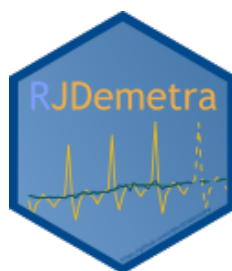


Table of contents

Introduction	1
1 R packages description	2
1.1 RJDemetra	2
1.2 rjwsacruncher	3
1.3 JDCruncherR	4
1.4 rjdworkspace	5
1.5 rjdqa	5
1.6 rjdmarkdown	5
1.7 ggdemetra	6
2 Seasonal Adjustment in R	6
2.1 Adjusting series with X13-Arima or Tramo-Seats	6
2.2 Output organization	7
2.2.1 Retrieving series	7
2.2.2 Retrieving diagnostics	8
2.3 Data visualization	8
2.4 Customizing parameters	10
2.4.1 Creating a new specification	10
2.4.2 Key interventions	10
2.5 The user-defined output	12
2.6 The quality assessment dashboard	12
2.7 Graphics for communication	13
2.8 Presenting parameters and diagnostics	15
3 Time series analysis combining the JDemetra+ graphical interface and R packages	17
3.1 JDemetra+ graphical user interface	17
3.2 Seasonal adjustment detailed analysis	18
3.3 Generating and handling JDemetra+ data	19
3.4 Using the right tool	20
Conclusion	21
References	22

Introduction

The RJDemetra package [1] is an interface between R and JDemetra+ (JD+), the Eurostat recommended seasonal adjustment software [2], offering full access to the JD+ seasonal adjustment routines. Several add-in packages expand its features: rjdworkspace [3], rjdqa [4], rjdmarkdown [5] and ggdemetra [6].

JDemetra+ is an open source software for seasonal adjustment (SA), outlier detection, time series modeling, nowcasting and temporal dis-aggregation. It has been officially recommended by Eurostat to the European Statistical System members since 2015. For seasonal adjustment, our focus here, JDemetra+ implements the concepts and algorithms used in the two currently leading methods worldwide: Tramo-Seats [7] and X13-Arima-Seats [8]. It provides the user with a rich graphical interface facilitating quality assessment, as well as the cruncher, a production module designed for batch treatments.

This paper aims at describing and illustrating the functionalities of those packages, presented in part one, used stand-alone, as detailed in part 2, or in conjunction with the JD+ graphical interface, as outlined in part 3. It also presents an older staple tool for production: the jwsacruncher package.

Two kinds of audiences can benefit from those tools: JDemetra+ users seeking extended functionalities in R, and R users not necessarily familiar with the JDemetra+ software, but willing to perform reliable seasonal adjustment using officially recommended algorithms. The latter should focus on part two of this document.

Basic concepts related to seasonal adjustment are not exposed in this document. Interested readers can refer to the online JDemetra+ documentation, which will also provide additional references. Among the two seasonal adjustment algorithms available in the JDemetra+ software, we have chosen to base our examples on X13-Arima. The ground structure of the available functions is very similar when using Tramo-Seats. Seasonal adjustment with Tramo-Seats is quasi-identical for the pre-processing stage, but requires different parameters and yields different outputs in the decomposition phase. The RJDemetra documentation will help.

1 R packages description

The recent R interface for JDemetra+ made the X13-Arima and Tramo-Seats algorithms directly available in R, as well as the JDemetra+ rich output (series, specifications and diagnostics). The core package, RJDemetra, was released in 2018. It has since been enriched with several add-ins: rjdworkspace for data processing, rjdqa for quality assessment, ggdemetra for seasonal adjustment specific graphics with ggplot2 and rjdmarkdown for printing SA processing features in Latex or HTML format.

In this section, we first cover the "production packages": RJDemetra, jwsacruncher and JDCruncher, enabling to refresh data and generate a quality report; as well as rjdworkspace, containing functions to "mix and match" series and metadata.

We then present packages useful for reporting: rjdqa, generating a dashboard summarizing the main parameters and quality indicators of a series adjustment; rjdmarkdown, simplifying the inclusion of results and diagnostics in reports; and finally ggdemetra, creating informative graphs.

From a technical standpoint: since RJDemetra relies on the rJava package that requires at least Java SE 8, all following packages require at least that version too.

1.1 RJDemetra

Main objectives

RJDemetra enables the use of the JDemetra+ Java routines in R, offering full access to both X13-Arima and Tramo-Seats methods, as well as to the reg-Arima and Tramo pre-adjustment steps and all the options and outputs of JDemetra+.

Installation procedure

The package can be downloaded from a CRAN mirror using the usual command:

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

In case of error, refer to the installation manual: <https://github.com/jdemetra/rjdemetra/wiki/Installation-manual>

Main functionalities

With the RJDemetra package, the user can load a pre-existing workspace (created via the JDemetra+ interface, for example), access and modify its components (multiprocessings, series and their specifications) before saving it all. A workspace can also be created from scratch with manual specifications.

A JDemetra+ output contains three kinds of objects: series (input and output), specifications (a set of parameters defining the seasonal adjustment process) and diagnostics (a set of values and test suitable for quality assessment). In RJDemetra, this output is organized in five main lists, detailed in sub-lists:

```
SA
├── reg-Arima (≠ X13 and Tramo-Seats)
│   ├── specification
│   └── ...
├── decomposition (≠ X13 and Tramo-Seats)
│   ├── specification
│   └── ...
├── final
│   ├── series
│   └── forecasts
├── diagnostics
│   ├── variance_decomposition
│   ├── combined_test
│   └── ...
└── user_defined
```

Since RJDemetra calls Java routines, there are two ways to manipulate an object (e.g. a workspace, a SAP, an sa_item, a model, etc.): the user can use transcribed "normal R objects" or Java objects. While the latter is less user-friendly, it also reduces drastically the computing time.

The RJDemetra package gives access to every diagnostic computed by JDemetra+, which can then be used to calculate a customized quality score. However, it doesn't allow to automatically refresh seasonally adjusted data according to a chosen revision policy. Therefore, the RJDemetra package is not

meant for infra-annual production: this operation requires to use the graphical interface or either package `jwsacrunner` or `JDcruncher`. Nevertheless, `RJDemetra` can be used after the output has been updated, to automatically generate reports containing month-to-month evolution rates and outlier changes, for example.

The list of all `RJDemetra` functions can be found on its R help page.

Example

To seasonally adjust a time series with a pre-defined specification, the user can either use the `x13()` function for the X13-Arima method or the `tramoseats()` function for the Tramo-Seats method:

```
library(RJDemetra)
# The RJDemetra package encapsulates the aggregated Industrial Production Index (IPI) series
# by country.
myseries <- ipi_c_eu[, "FR"]
# Seasonal adjustment using the default methods
x13_result <- x13(myseries)
ts_result <- tramoseats(myseries)
```

The seasonal adjustment processed will be detailed in part 2.

Thanks to the function `load_workspace()`, the user can also retrieve series and their characteristics previously defined via the `JDemetra+` graphical interface. Further and more detailed operations with `RJDemetra` are presented in part 3.

1.2 `rjwsacrunner`

Main objectives

The `rjwsacrunner` package allows the user to update a workspace and to export the results from the R console, without having to open `JDemetra+`. During the process, all multi-processings defined in the workspace are re-estimated according to the chosen revision policy. Only one policy can be applied to a given workspace and all its multi-processings.

The refresh policies parameters are listed here: <https://github.com/jdemetra/jwsacrunner/wiki>.

Installation procedure

The package can be downloaded from a CRAN mirror using the usual command:

```
install.packages("rjwsacrunner")
```

Warning: the cruncher in itself is not included in the package. It can be downloaded on Github (<https://github.com/jdemetra/jwsacrunner/releases>) or via the following command:

```
# Directory where to save the JWSACruncher:
directory <- "D:/"
download_cruncher(directory)
```

For further help on the cruncher: <https://github.com/jdemetra/jwsacrunner/wiki>.

Example

To perform the concurrent refreshment of a workspace:

```
library(rjwsacrunner)
# Path to the "bin" directory
options(cruncher_bin_directory = "D:/jdemetra-cli-2.2.3/bin/")

cruncher_and_param(workspace = "my_ws.xml",
  rename_multi_documents = FALSE, # to rename the output folder
  delete_existing_file = TRUE, # to override the existing files
  policy = "complete", # the refresh policy
  csv_layout = "vtable", # the output format
  log_file = "my_folder/my_ws_log.txt")
```

Exported items can also be changed:

```
# To get the default values:
head(getOption("default_matrix_item"))
#> [1] "period" "span.start" "span.end" "span.n" "span.missing" "espan.start"
getOption("default_tsmatrix_series")
#> [1] "y" "t" "sa" "s" "i" "ycal"
# To only export the seasonally adjusted series and its forecasts:
options(default_tsmatrix_series = c("sa", "sa_f"))
```

1.3 JDCruncher

Main objectives

JDCruncher is another package calling the cruncher and allowing the user to update a workspace and export the results without having to open JDemetra+. On top of that, JDCruncher enables the calculation of a quality score which can be customized by selecting the contributing variables and choosing their weights.

The refresh policies parameters can be found here: <https://github.com/jdemetra/jwsacruncher/wiki>.

Installation procedure

The package can be downloaded from the Insee github: <https://github.com/InseeFr/JDCruncher>.

Warning: the cruncher is not included in the package or in the Insee Github. To download it, refer to the jwsacruncher section.

Example

```
library(JDCruncher)
# Updating the workspace with the policy "last outlier"
cruncher_and_param(workspace = "my_path/my_workspace.xml",
                    policy = "current",
                    csv_layout = "vtable", # for the cvs format
                    log_file = "my_workspace/log.txt")
```

By default, the output tables will be placed in a new "Output" subfolder inside the workspace folder. If said folder already exists, the tables will be updated. If needed, another path and folder can be chosen by adding the argument 'output="another_path/another_folder"'.

To calculate a series score:

```
# To import of the diagnostics matrix
QR_matrix <- extract_QR("my_path/my_workspace/output/SAProcessing-1/demetra_m.csv")
# To calculate the score with the default settings
QR <- compute_score(QR_matrix)
# To calculate a weighted score
QR_w <- compute_score(QR_matrix, "pond")
```

The resulting QR_matrix (Quality Report) is a list of three items:

- the data frame *modalities*, containing the quality level of a number of variables (Good, Uncertain, Bad and Severe, by default);
- the data frame *values*, containing the actual values of the indicators in the *modalities* data frame (e.g.: p-values, statistics, etc.) as well as non modal variables (e.g.: series frequency, arima model, etc.);
- the *score_formula*, once a score has actually been calculated.

1.4 rjdworkspace

Main objectives

While RJDemetra enables the user to manipulate parameters to adjust the series, rjdworkspace enables the user to manipulate the series themselves and the associated comments: some of the functionalities include removing a series from a multiprocessing, replacing it, as well as changing the raw data of a series while keeping its model. The package also enables to extract and update the comments of a series or a workspace.

Installation procedure

The package can be downloaded from the Insee Lab github: <https://github.com/InseeFrLab/rjdworkspace>.

Main functionalities

The main functions of the package are:

- `add_new_sa_item()`, `remove_sa_item()` and `replace_sa_item()` which allow to "mix and match" parts of workspaces;
- `set_spec()` and `set_metadata()` which enable to attribute to a series a whole given spec and even more metadata elements (comments, the path to the raw data file, etc.);
- `merge_ws()` and `update_metadata()`, which respectively generalize the `set_metadata()` function to a chosen number of series and to a whole workspace.

They can be particularly useful during annual campaigns.

Example

Examples and more details are provided in part 3.3.

1.5 rjdqa

Main objectives

The rjdqa produces dashboards summarizing single series seasonal adjustment results.

Installation procedure

The package can be downloaded from a CRAN mirror using the usual command:

```
install.packages("rjdqa")
```

Example

An example is provided in section 2.6

1.6 rjdmarkdown

Main objectives

rjdmarkdown contains a set of functions for curated printouts of parameters or diagnostics from a seasonal adjustment process.

Installation procedure

The package can be downloaded from a CRAN mirror using the usual command:

```
install.packages("rjdmarkdown")
```

An example is provided in section 2.8.

1.7 ggdemetra

Main objectives

ggdemetra is a tool to gather series plots and seasonal adjustment information on the same figure, mainly for reports and presentations. The package is an extension of ggplot2. It calls the RJDemetra package to seasonally adjust the series and retrieve the model information (outliers, Arima orders).

Installation procedure

The package can be downloaded from a CRAN mirror using the usual command:

```
install.packages("ggdemetra")
```

Main functionalities

The four main functions aim at adding components of the adjustment process to a graph or to display information on different pre-adjustment components on or around said graph:

- `geom_sa()` adds a series component such as the trend, the seasonally adjusted time series, forecast values, etc.;
- `geom_outlier()` displays the outliers used in the pre-adjustment phase;
- `geom_arima()` displays the arima model used in the pre-adjustment phase;
- `geom_diagnostics()` adds a table containing diagnostics on the seasonal adjustment process.

A detailed example is provided in part 2.

2 Seasonal Adjustment in R

The R user can perform seasonal adjustment with Tramo-Seats or X13-Arima with a pre-defined or a user-defined specification, including user-defined calendar regressors and intervention variables. Specification modification is available in a “save as” manner from any existing specification. Said specification may have previously been defined in the JDemetra+ graphical user interface or in R. The latter makes massive parameters adjustments easier for simulation or production purposes and the ggdemetra package, built on the ggplot2 package, allows to replicate and customize the JDemetra+ graphics style.

2.1 Adjusting series with X13-Arima or Tramo-Seats

The RJDemetra seasonal adjustment function has 2 arguments: a raw series in the format of a time series object, and a specification. Just as JDemetra+ offers a set of pre-defined (working days) specifications to start with, RJDemetra has such options and their syntax is detailed in the package help page.

The result of the SA function is referred to as a model, which is a formatted output organized in lists and sub-lists, containing series, parameters and diagnostics, as described in section 1.1.

```
# Creation of a time series object from a data frame
raw_series<-ts(ipi[, "RF3030"], frequency=12, start=c(1990,1), end=c(2019,9))

# Seasonal adjustment with X13 or Tramo-Seats
model_sa<-x13(raw_series, spec = "RSA5c")
model_sa_ts<-tramoseats(raw_series, spec = "RSAfull")

# To only perform the pre-adjustment step: the reg-Arima part
my_reg <- regarima_x13(raw_series, spec = "RG5c")
summary(my_reg)
```

In the following section, we base our description and examples upon the X13 algorithm and run a full seasonal adjustment process. As seen above, it is also possible to only perform the pre-adjustment (ie. the reg-Arima part).

2.2 Output organization

Once the seasonal adjustment has been computed, it's time to access the output: the series, the parameters and the diagnostics.

The output is the same as with the JD+ software. For the comprehensive list of exportable items, check: <https://jdemetradocumentation.github.io/JDemetra-documentation/pages/theory/output.html>.

2.2.1 Retrieving series

The *final* sub-list contains two lists of time series objects: the final components and their forecasts (with the "_f" suffix). Said components are the following:

- y: raw series
- sa: seasonally adjusted series
- s: seasonal component
- t: trend-cycle
- i: irregular.

```
# To retrieve final components (y, sa, s, t, and i)...
ts_final <- model_sa$final$series
# ...and their forecasts y_f, sa_f, s_f, t_f, and i_f
ts_final_forecasts <- model_sa$final$forecasts
```

The series calculated during the pre-processing step can also be retrieved, as described below. Note that if the chosen decomposition model happens to be multiplicative, the displayed linearised series is actually its log. The default available components are the following:

- y_lin: linearized series
- tde: total trading days effect
- ee: Easter effect
- omhe: other moving holidays effect
- out_t: outlier effect on trend
- out_s: outlier effect on the seasonal component
- out_i: outlier effect on the irregular
- out: total outlier effect.

For more options, check the user-defined output section.

```
# To retrieve the default pre-processing components (y_lin , tde, ee, omhe, out_t, out_s,
# out_i and out)
ts_preprocessing <- model_sa$regarima$model$effects

# To revert to y_lin from its log value (displayed if the model is multiplicative)
schema <- model_sa$regarima$model$spec_rslt[,3]
# To retrieve the pre-processing series
ts_preprocessing <- model_sa$regarima$model$effects
# To correct the linearized series (first column of the pre-processing vector)
ifelse(schema,ts_preprocessing[,1] <- exp(ts_preprocessing[,1]),ts_preprocessing[,1])
```

Finally, evaluating the pre-processing quality might require checking the forecasts residuals:

```
# To retrieve forecasts residuals
ts_residual_forecasts <- model_sa$regarima$forecast
```

2.2.2 Retrieving diagnostics

Assessing the quality of the seasonal adjustment process requires retrieving diagnostics. Below are examples of retrieval code for some of the main diagnostics. Some of the resulting outputs are displayed in section 2.8, related to the `rjdmardown` package.

```
# Regression variables and coefficients
# Relative to all deterministic effects, both automatic and used-defined:
# trading days, Easter, outliers and more. For an example of output, see section 2.8 Table 2)
model_sa$regarima$regression.coefficients

# Arima coefficients (see section 2.8 Table 1)
model_sa$regarima$arima.coefficients

# Statistics on residuals (see section 2.8 Table 5)
model_sa$regarima$residuals.stat$tests

# Decomposition: the M-stats (see section 2.8 Table 3)
model_sa$decomposition$mstats

# Final Diagnostics
# Test for the presence of seasonality (X13)
model_sa$diagnostics$combined_test
# Test for residual seasonality and residual trading days on the seasonally adjusted series
# and the irregular
model_sa$diagnostics$residuals_test
# X13 variance decomposition
model_sa$diagnostics$variance_decomposition
```

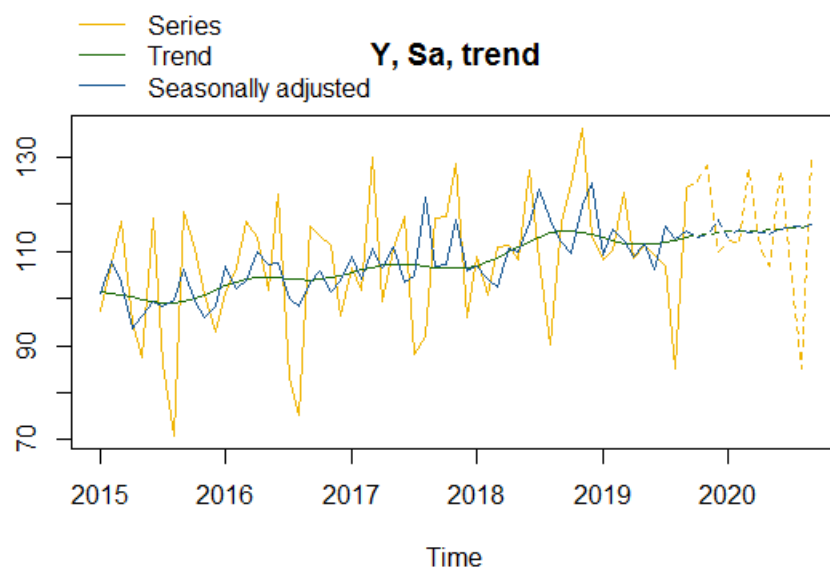
2.3 Data visualization

RJDemetra contains built-in plot functions, specifically for seasonal adjustment.

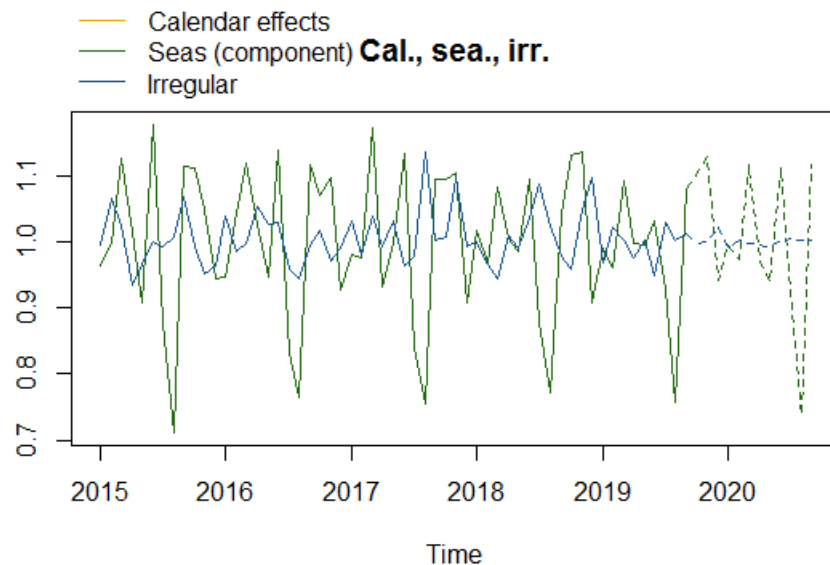
Plotting final components

For the final series components, two types of plots are available:

```
# The raw series, seasonally adjusted series and trend
plot(model_sa, type_chart = "sa-trend", first_date = c(2015, 1))
```



```
# The calendar and seasonal effects + the irregular component
plot(model_sa, type= "cal-seas-irr", first_date = c(2015, 1))
```

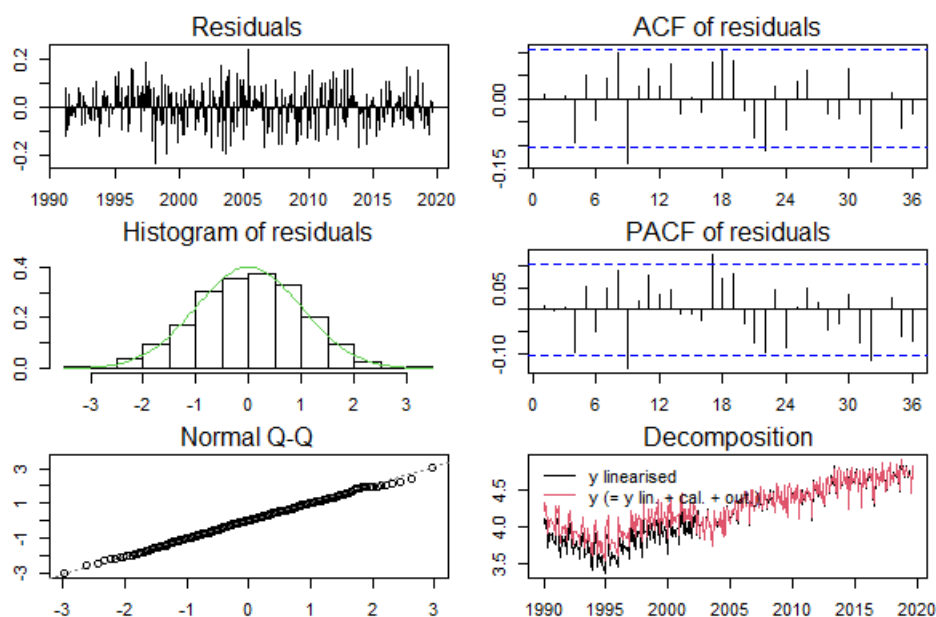


The pre-adjustment step

For the pre-adjustment part, six types of graphics are available:

```
# Template
plot(model_sa$regarima,
      which = 1:6,
      caption = list("Residuals", "Histogram of residuals", "Normal Q-Q",
                    "ACF of residuals", "PACF of residuals", "Decomposition",
                    list("Y linearized",
                        "Calendar effects", "Outliers effects"))[sort(which)], ask=FALSE)

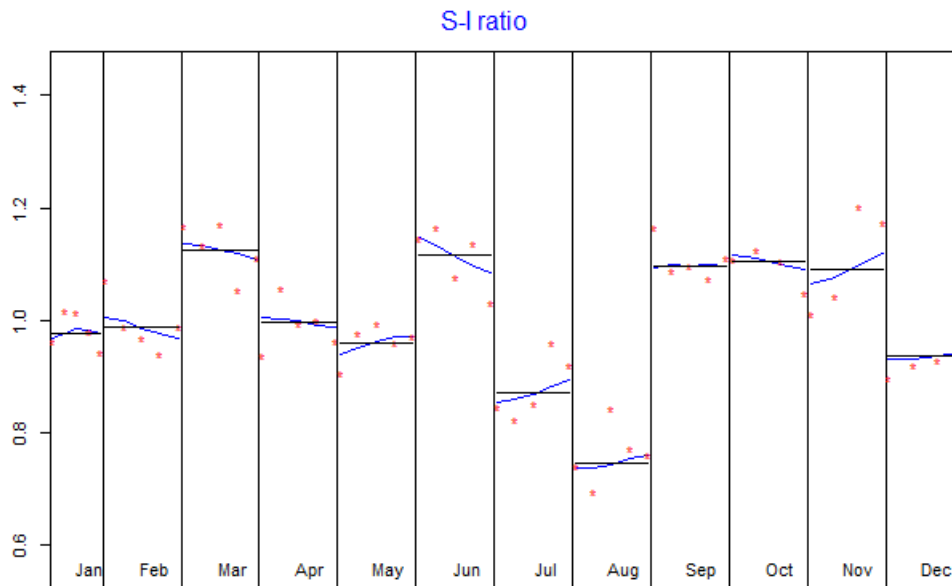
# To print all six plots at once
layout(matrix(1:6, 3, 2)); plot(model_sa$regarima, ask = FALSE)
```



The decomposition step

For the X13 decomposition part, one type of plot is available: it displays the so called 'S-I ratios'. They represent the evolution of the seasonal factors (in blue) for each period type (month or quarter for now), the black straight line being their average and the red dots, $S+I$ or $S*I$.

```
# Template
plot(x, first_date, last_date, caption = "S-I ratio", ylim,...)
# Plotting SI ratios
plot(model_sa$decomposition, first_date = c(2015,1))
```



2.4 Customizing parameters

2.4.1 Creating a new specification

A specification is the set of parameters the algorithm needs to run the seasonal adjustment process. In absence of input from the user, it will fall back on one of the pre-defined specifications. The latter are listed in the package help page. X13-Arima and Tramo-Seats have indeed several default specifications, differing mostly by the selected arima model and trading days regressors.

Manually writing a specification from scratch is possible, although really burdensome (see the RJDemetra help page in R for all the parameters to fill in). Fortunately, creating a user-defined specification can be done by adapting any existing specification, in a "save as" manner. It is common to start with a pre-defined specification and modify what is of interest. Below is an example of the recovery of a pre-existing specification.

```
# To store the specification corresponding to a previous adjustment
# spec_1 will correspond to RSA5c, as specified in section 2.1 where "model_sa" was defined
spec_1<- x13_spec(model_sa)
```

Its modification on selected points is the subject of the next sub-section.

2.4.2 Key interventions

Some of the most common manual interventions include:

- changing the estimation span (only for the pre-adjustment, not the decomposition)
- imposing a multiplicative model

- adding outliers
- defining the arima model
- adding user-defined trading-day regressors
- changing decomposition filters for X13.

Example 1

```
# To change the estimation span, impose an additive model and add user-defined outliers
# First, we create a new specification by modifying the previous one
spec_2 <- x13_spec(spec_1, estimate.from = "2004-01-01",
  usrdef.outliersEnabled = TRUE,
  usrdef.outliersType = c("LS", "AO"),
  usrdef.outliersDate = c("2008-10-01", "2018-01-01"),
  transform.function = "None") # forcing the additive model
# Here, the reg-arima model will be estimated from "2004-01-01" but the decomposition will be
# run on the whole span.
# Then, we create the new SA processing with this new specification
model_sa_2 <- x13(raw_series, spec_2)
```

Example 2

```
# To change the arima model, starting from spec_1
# Here, we force a (1,d,1)(0,D,1).
# The regular coefficients are fixed: to -0.8 for the AR(1) and -0.6 for the MA(1).
# The seasonal AR order is set to 0 and the coefficient for the seasonal MA(1) is not fixed
# by the user ("Undefined")
spec_3 <- x13_spec(spec_1, automdl.enabled = FALSE,
  arima.p = 1, arima.q = 1,
  arima.bp = 0, arima.bq = 1,
  arima.coefEnabled = TRUE,
  arima.coef = c(-0.8, -0.6, 0), # 0 stands for not fixed
  arima.coefType = c(rep("Fixed", 2), "Undefined"))
# We then create the new SA processing with this customized specification
model_sa_3 <- x13(raw_series, spec_3)

# To access the specification parameters...
s_arimaCoef(spec_3)
# ... and the estimation results for the arima model
model_sa_3$regarima$arima.coefficients
```

Example 3

```
# To specify user-defined trading days
spec_4 <- x13_spec(spec_1,
  tradingdays.option = "UserDefined",
  tradingdays.test = "None",
  usrdef.varEnabled = TRUE,
  usrdef.varType = "Calendar", # the user-defined variable will be assigned to the calendar component
  usrdef.var = regs_cjo) # the regressors must be in the time series format
# Creation of a new SA processing
model_sa_4 <- x13(raw_series, spec_4)
# To retrieve the results
model_sa_4$regarima$regression.coefficients
```

Example 4

```
# To specify an intervention variable
spec_5 <- x13_spec(spec_1,
```

```

usrdef.varEnabled = TRUE,
usrdef.varType="Trend", # the variable will be assigned to the trend component
usrdef.var=x) # x must be in the time series format
# The new SA processing
model_sa_5<-x13(raw_series,spec_5)
# Results recovery
model_sa_5$regarima$regression.coefficients

```

Example 5

```

# To customize decomposition filters:
# - the Henderson moving average (ma) length for the trend
# - and the Seasonal moving average (ma) length for seasonal component's extraction
# Starting from spec_1
spec_6 <- x13_spec(spec_1, x11.trendma = 23, # the Henderson filter length is set to 23 periods
  x11.seasonalma = "S3X9") # the seasonal filter is a 3X9 moving average
# Creating the new SA processing
model_sa_6 <- x13(raw_series, spec_6)

```

After customization, a specification can be saved in and loaded back from an Rdata file:

```

# The save_spec function
save_spec(spec_3, file.path("my_path","spec_3_user_def_arima.RData"))
# The load_spec function
new_spec<-load_spec(file="my_path/spec_3_user_def_arima.RData")

```

2.5 The user-defined output

Additional series and diagnostics can be retrieved from the "user-defined" sub-list generated by the seasonal adjustment function. The full list is in the documentation:

<https://jdemetradocumentation.github.io/JDemetra-documentation/pages/theory/output.html>

```

# To print the list of user-defined items recoverable in the output
user_defined_variables("X13-Arima")
# When stored into a variable, it is in the format of a character vector.

# Example: selecting the forecast of the linearized series (y_lin_f) and the D1 component
# of the X13 decomposition. (They happen to be the 28th and 174th items of the list.)
my_output <- user_defined_variables("X13-Arima")[28,174]

# Running the SA function with the user-defined output
model_sa <- x13(raw_series, spec ="RSA5c", userdefined = my_output)

# Retrieving the list of time series objects
additional_series <- model_sa$user_defined

```

2.6 The quality assessment dashboard

The rjdqa add-in can readily help set up a quality assessment report using JDemetra+ diagnostics. For a single-series detailed analysis, it provides a convenient dashboard.

```

library(RJDemetra)
library(rjdqa)
# The function argument is the result of a seasonal adjustment
dashboard_data <- sa_dashboard(model_sa)
plot(dashboard_data, main = "Dashboard for series RF3030",
  subtitle = "SA with X13")
# To display the diagnostics in the R console
dashboard_data$summary_diagnostics

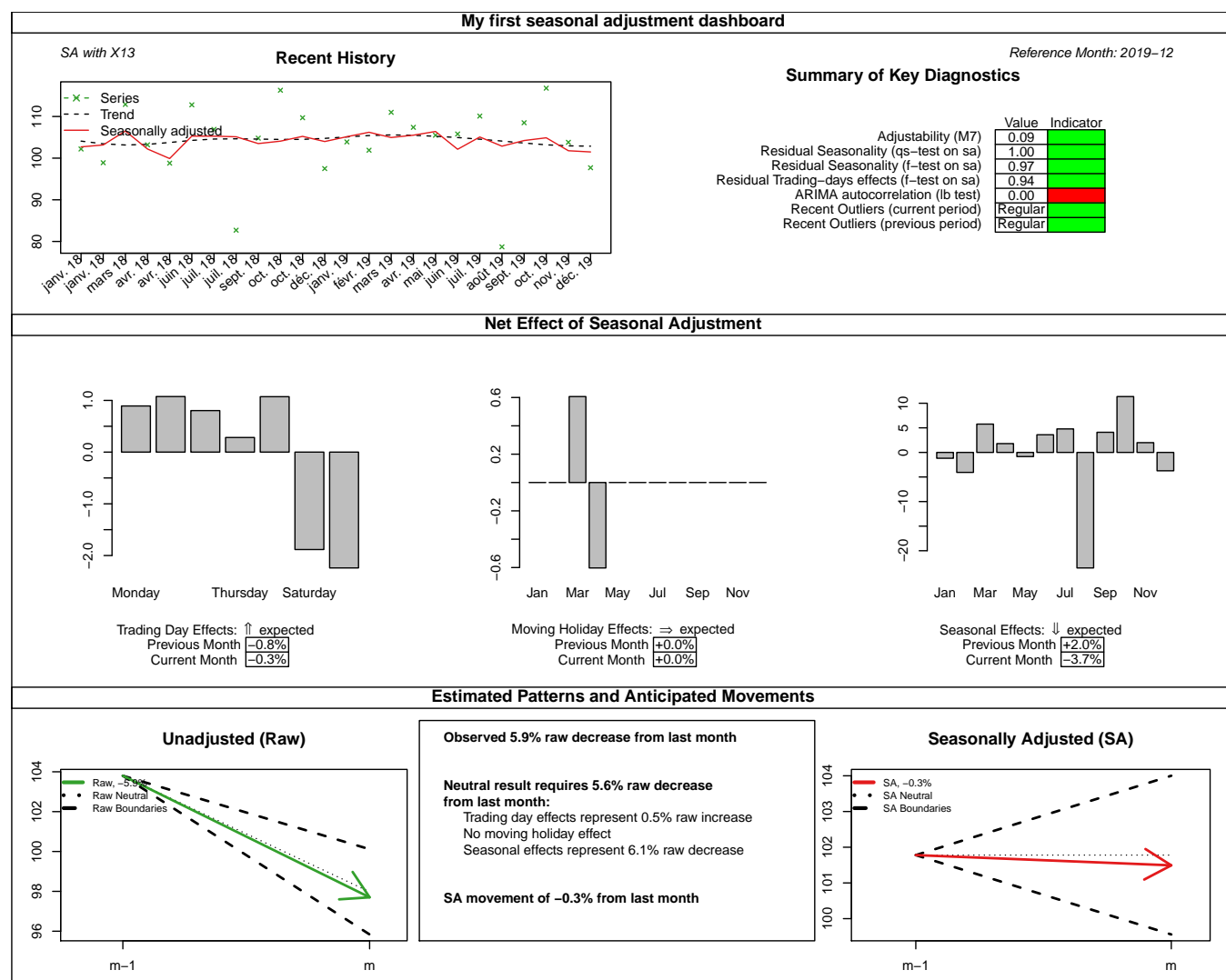
```


Details can be retrieved via the `dashboard_data` list.

```
# For example, to display the seasonal factors values
dashboard_data$seasonal_pattern
# To display the dashboard when working under Rmarkdown
```{r, fig.height = 8, fig.width = 10}
library(rjtdqa)
data <- window(RJDemetra::ipi_c_eu[, "FR"], start = 2003)
sa_model <- RJDemetra::x13(data, "RSA5c")
dashboard_data <- sa_dashboard(sa_model)
plot(dashboard_data, main = "French Industrial Production Index",
 subtitle = "SA with X13")
...

And to export it in png format
```{r, fig.height = 8, fig.width = 10, fig.ext='png', fig.cap="Png figure"}
plot(dashboard_data, main = "French Industrial Production Index",
      subtitle = "SA with X13")
...

```



2.7 Graphics for communication

ggemetra allows adding information about the seasonal adjustment process to a ggplot2 graphic. Said additional information, contained in a specification ("spec"), can be:

- a specification generated in an SA processing previously created with RJDemetra, or created with JD+ and imported and computed with RJDemetra
- a specification defined on its own without even actually running a SA process
- a default specification.

Four types of information can be added:

- components (series)
- outliers
- arima model characteristics
- diagnostic table.

```
# Classical plot of the raw series using the ggplot2 package
# The data must be a data frame with a numerical date format. The conversion from time series
# format to numerical date can be done with the ts2df function, included in the ggdemetra package.
```

```
base_plot<- ggplot(data = ipi_df, mapping = aes(x = date, y = RF2740)) +
  geom_line() +
  labs(title = "Industrial Production Index (IPI)",
       x = "date", y = "RF2740")
base_plot
```

```
# Defining a specification...
```

```
spec <- RJDemetra::x13_spec("RSA3", tradingdays.option = "WorkingDays")
```

```
# ... plotting the augmented graph
```

```
enhanced_plot<- base_plot+
```

```
# ... adding the adjusted series and both forecasts to the plot, with the "geom_sa" function
```

```
  geom_sa(component = "y_f", linetype = 2,
         spec = spec) +
  geom_sa(component = "sa", color = "red") +
  geom_sa(component = "sa_f", color = "red", linetype = 2)+
# as well as pinpointing outliers and displaying their dates
  geom_outlier(geom = "label_repel",
              vjust = 4,
              ylim = c(NA, 65), force = 10,
              arrow = arrow(length = unit(0.03, "npc"),
                           type = "closed", ends = "last"))+
```

```
# ... and the arima model orders
```

```
  geom_arima(geom = "label",
            x_arima = - Inf, y_arima = -Inf,
            vjust = -1, hjust = -0.1,
            message = FALSE)
```

To add a diagnostics table to the figure, the user must list the desired diagnostics using their name from the output file (e.g. diagnostics.combined.all.summary). However, said diagnostics can be renamed so that a more readable label appears on the graph (e.g. Seasonality (combined)).

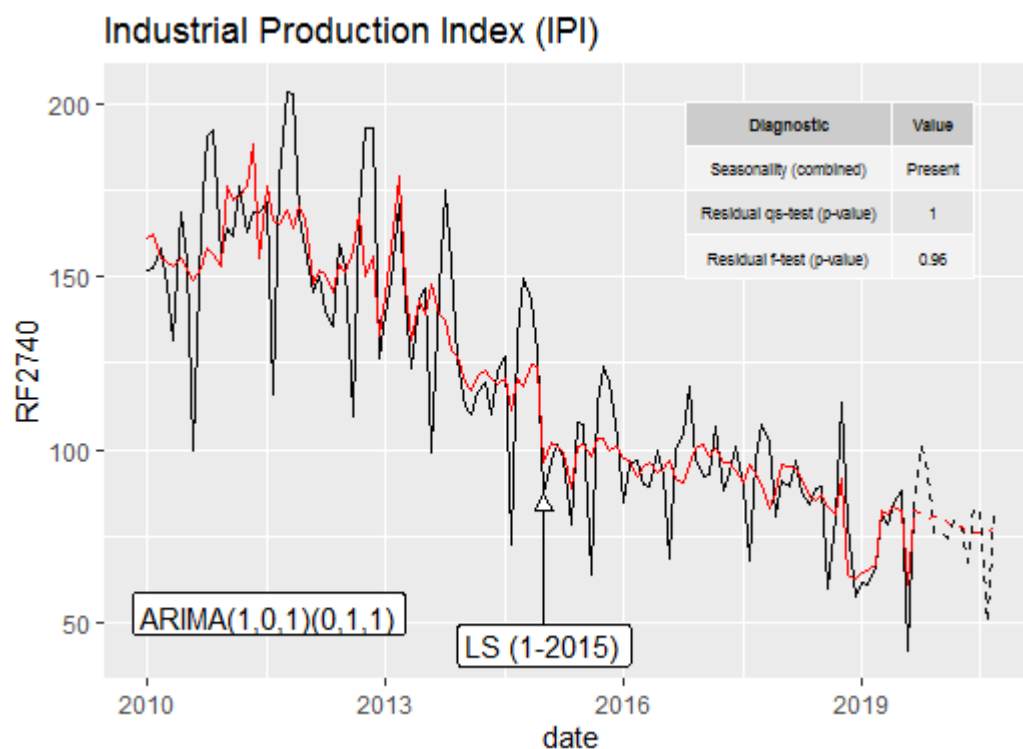
```
# The vector of renamed diagnostics
```

```
diagnostics <- c(`Seasonality` (combined)` = "diagnostics.combined.all.summary",
                `Residual qs-test` (p-value)` = "diagnostics.qs",
                `Residual f-test` (p-value)` = "diagnostics.ftest")
```

```
# Addition to the figure
```

```
enhanced_plot+ geom_diagnostics(diagnostics = diagnostics,
                              ymin = 150, ymax = 200, xmin = 2016,
                              table_theme = gridExtra::ttheme_default(base_size = 6))
print(enhanced_plot)
```

The final enhanced plot



2.8 Presenting parameters and diagnostics

Relevant parameters or diagnostics from the seasonal adjustment process can be printed out in a \LaTeX or html report, using `rjdmardown`:

```
library(RJDemetra)
library(rjdmardown)

print_preprocessing(model_sa, format="latex")
```

Pre-processing (RegArima)

Summary

357 observations
 Series has been log-transformed
 Trading days effect (6 variables)
 Easter [8] detected
 1 detected outlier

Likelihood statistics

Number of effective observations = 344
 Number of estimated parameters = 11
 Log-likelihood = 386.141, AICc = 2183.342, BICc = -4.929
 Standard error of the regression (ML estimate) = 0.078

Arima model

Table 1: Arima coefficients

	Coefficients	Std. Error	T-stat	$\mathbb{P}(> t)$	
Theta(1)	-0.767	0.035	-21.745	0.000	***
BTheta(1)	-0.564	0.046	-12.187	0.000	***
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1					
Arima (0,1,1)(0,1,1)					

Regression model

Table 2: Regression coefficients

	Coefficients	Std. Error	T-stat	$\mathbb{P}(> t)$	
Monday	0.017	0.009	1.930	0.054	.
Tuesday	0.011	0.009	1.225	0.222	
Wednesday	0.009	0.009	1.025	0.306	
Thursday	-0.003	0.009	-0.409	0.683	
Friday	0.020	0.009	2.273	0.024	*
Saturday	-0.021	0.009	-2.449	0.015	*
Easter [8]	-0.045	0.017	-2.621	0.009	**
LS (5-2002)	-0.175	0.046	-3.840	0.000	***
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1					

```
print_decomposition(model_sa)
```

Decomposition (X13) Mode: multiplicative

Table 3: M-statistics

	Value	Description
M-1	0.977	The relative contribution of the irregular over three months span
M-2	0.687	The relative contribution of the irregular component to the stationary portion of the variance
M-3	1.977	The amount of period to period change in the irregular component as compared to the amount of period to period change in the trend
M-4	0.407	The amount of autocorrelation in the irregular as described by the average duration of run
M-5	3.000	The number of periods it takes the change in the trend to surpass the amount of change in the irregular
M-6	0.111	The amount of year to year change in the irregular as compared to the amount of year to year change in the seasonal
M-7	0.372	The amount of moving seasonality present relative to the amount of stable seasonality
M-8	0.824	The size of the fluctuations in the seasonal component throughout the whole series
M-9	0.428	The average linear movement in the seasonal component throughout the whole series
M-10	0.589	The size of the fluctuations in the seasonal component in the recent years
M-11	0.499	The average linear movement in the seasonal component in the recent years
Q	0.943	
Q-M2	0.974	
Final filters: M3x5, Henderson-23 terms		

Table 4: Relative contribution of the components to the stationary portion of the variance in the original series, after the removal of the long term trend

Component	
Cycle	7.941
Seasonal	50.203
Irregular	9.042
TD & Hol.	2.782
Others	28.402
Total	98.370

```
print_diagnostics(model_sa)
```

Table 5: Diagnostics tests

	$\mathbb{P}(> t)$	
mean	0.709	
skewness	0.503	
kurtosis	0.593	
ljung box	0.013	*
ljung box (residuals at seasonal lags)	0.875	
ljung box (squared residuals)	0.027	*
qs test on sa	1.000	
qs test on i	1.000	
f-test on sa (seasonal dummies)	0.993	
f-test on i (seasonal dummies)	0.995	
Residual seasonality (entire series)	0.984	
Residual seasonality (last 3 years)	0.776	
f-test on sa (td)	0.926	
f-test on i (td)	0.999	
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1		

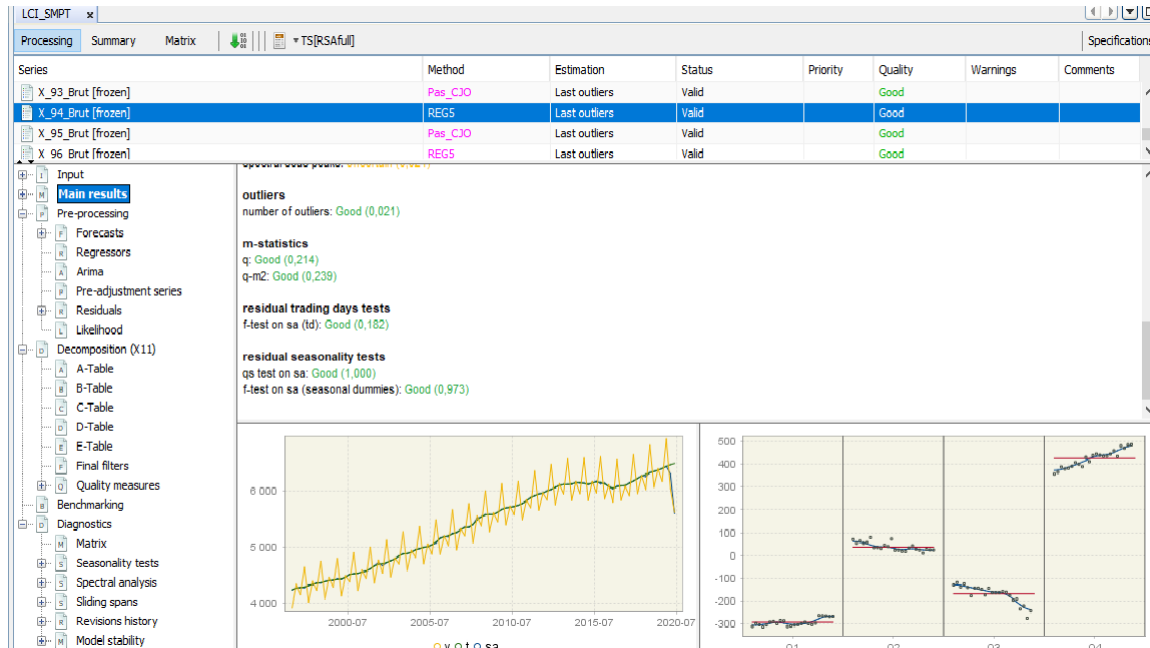
3 Time series analysis combining the JDemetra+ graphical interface and R packages

RJDemetra gives instant access to the entire output (series, specifications and diagnostics) available in JDemetra+. Monitoring and fine-tuning a SA process can be simultaneously done from the JDemetra+ graphical interface and R. The graphical interface offers a structured and visual feedback whereas R provides comparison facilities, not available within the interface. For example, the impact of parameters adjustment can be highlighted on the spot, without exporting and re-importing data. The parameters changes are suitably assessed on a macro and micro level: first spotting problematic revisions or specification shifts compared to a previous version of the data, then fine-tuning specification choices on selected series.

3.1 JDemetra+ graphical user interface

The JDemetra+ graphical user interface (GUI) is designed as a launchpad bringing together X13 and Tramo-Seats algorithms. A complete SA process can be run from the interface: importing raw data to adjust as well as auxiliary variables, defining global specifications for the entire series set, fine-tuning parameters for selected series and exporting results. A major asset of the GUI is its hierarchically organized output (series, plots, parameters, diagnostics), giving instant feedback on the process quality.

Said feedback has multiple layers (from global to very detailed ones) and can be read with increasing levels of scrutiny from colored hints to detailed p-values and distributions. Users of various levels of time series analysis practice and interest can navigate the interface according to their needs (quite the contrary of a "the user will get all the details at once no matter what they are looking for" kind of design).



In the GUI, any parameter adjustment is immediately taken into account and new results, displayed. However, previous results cannot be stored: this renders comparing several versions of a given adjustment process quite burdensome (which is unfortunate, since it turns out to be unavoidable when fine-tuning seasonal adjustment processes). Here's where R/JDemetra can prove to be extremely useful.

3.2 Seasonal adjustment detailed analysis

R/JDemetra enables the user to design R scripts to instantly extract and compare selected elements from two or more workspaces. During an analysis, it can also be convenient to store baseline parameters and series in a WS_0 workspace, and to replicate it as a WS_1 workspace which will be left open to modifications via the GUI.

While working on a series in the GUI, after modifying some parameters of interest, the user will just have to save the current workspace (WS_1), without having to close the JDemetra+ session and run an R script to assess the impact of the changes on the selected series.

Below is an example of an extraction of output elements: here the final components sa, s, t, i series. Once extracted and gathered in common R objects (e.g data frames), designing reproducible graphs and estimating indicators is straightforward.

```
# List of paths to the workspaces
my_workspaces <- c("my_path/WS_0.xml", "my_path/WS_01.xml")

# Extracting the series from each workspace of the list and merging them together
for(i in seq_along(my_workspaces)){
  # adding a suffix to differentiate the origin of the extracted series
  suffix <- paste0("_", my_workspaces[i])
  # loading and computing each workspace
  ws <- load_workspace(my_workspaces[i])
  R/JDemetra::compute(ws)
  # retrieving the series of each workspaces' first SA-processing
  sap <- get_object(ws, 1)
  series <- get_all_objects(sap)
  # choosing one (common) series in WS_0 and WS_1
  sa <- get_object(sap, which(names(series) == my_series))
  # Extracting its main components
  model_sa <- get_model(sa, ws)
  final_comp <- model_sa$final$series
  # if needed: transforming the multiple TS object into a data frame
```

```

# creating a date variable (!\ package zoo required to display the "usual" format YYYY-MM-DD)
d <- data.frame(date=as.Date(time(final_comp)))
# creating a data frame for each workspace = each index i in the loop
df<-as.data.frame(final_comp)
df<-cbind(d,df)
# adding suffixes (except to the date column)
names(df)[-1]<-paste0(names(df)[-1],suffix)
# merging the data frames
if(i==1){final_table<-d}
final_table<-merge(final_table,df,by="date", all=TRUE)
}

```

Such simple R scripts for extracting series, tests results and other parameters simultaneously help building custom quality reports containing both statistical diagnostics and synthesized information on data revisions. During the analysis process, it can be useful to generate a macro quality report first, spot the series deserving scrutiny and then extract more detailed information on a single series level, which is also of great help for data exploration and studies.

3.3 Generating and handling JDemetra+ data

RJDemetra allows creating, importing and exporting JDemetra+ workspaces. However, while RJDemetra enables the user to add a series to an existing workspace thanks to the `add_sa_item()` function, the `rjdworkspace` package goes much further in terms of series, multiprocessing and workspace manipulation. For instance, it allows to merge two workspaces: that is particularly interesting at the end of annual campaigns, when the producer needs to gather the final models in a single workspace. `rjdworkspace` also allows to update a series metadata (specification, path to the raw series file, comments) with the metadata from another series. The user can add or modify the series comment to notify the change.

```

# To remove the third series from an SAProcessing:
my_sap <- remove_sa_item(my_sap,3)

# To replace the first series by another one (from another workspace or custom made in R
# with RJDemetra)
my_sap <- replace_sa_item(my_sap, 1, replacement_series)

# To add a new series
my_sap <- add_new_sa_item(my_sap, added_series)

# To attribute the specification of a series to another series :
spec2 <- get_spec(series2)
series1 <- set_spec(series1, spec2)

# To change a series specification and all its metadata (for example if the raw data
# has to be read from a different file)
series <- set_metadata(series, series_from_another_workspace)

# On a larger scale, to apply the specifications and metadata of a number of series to others,
# or even from a whole workspace to another one (for example, when comparing a workspace of
# reference to a test one):
merge_ws(ws_reference, ws_test, sap_name, series_list, path_final_ws.xml)
update_metadata(path_reference_ws.xml, path_test_ws.xml, path_new_ws.xml)

```

Notes:

- while the `merge_ws()` uses already loaded and computed workspaces, the `update_metadata()` function uses xml files paths as arguments.
- While the `set_spec()` function only changes specification parameters, both `set_metadata()`, `merge_ws()` and `update_metadata()` functions replace the whole series block in the xml file of the `SAProcessing`, including the associated raw data and eventual comments.

```
# To change the first series name and add a comment without changing any other metadata item
my_series <- get_objects(my_sap,1)
set_name(my_series,"first_series")
set_comment(my_series,"name change")
get_comment(my_series)
```

```
# To change the time series associated to a "series object":
```

```
my_series <- set_ts(my_series, other_ts)
```

rjdworkspace will prove useful for producers dealing with large data sets which have to undergo a number of automatic transformations, right before fine-tuning a limited number of series.

3.4 Using the right tool

The complementary nature of the set of tools presented in this paper, with the graphical user interface (GUI), should be underlined here.

The GUI allows to run all required operations for seasonal adjustment. Its best asset is the multi-layer user-feedback, very convenient for detailed analysis. However, its major shortcoming is the inability to store and simultaneously display previous results, preventing the user from easily visualizing the impact of a parameter change between two workspaces. RJDemetra will fill in this blank, making comparisons easy, and also provide a flexible framework for trying out a high number of specifications, and even two different algorithms, on a larger scale.

For mass production in batch mode, when data needs to be refreshed while keeping the specifications mostly fixed, the cruncher becomes necessary as RJDemetra+ doesn't handle revision policies yet. However, it might be the case in future versions linked to the upcoming JDemetra+ 3.0.

Conclusion

RJDemetra offers full access to the JDemetra+ seasonal adjustment routines and its add-in packages provide the user with convenient additional functionalities. A broader tool selection is henceforth available for seasonal adjustment: the JDemetra+ graphical interface, the cruncher and all R packages can be used in conjunction for greater benefit. In addition to making seasonal adjustment quality assessment easier, RJDemetra is also well suited for simulations, allowing to design massive specification adjustments within the flexible framework of an R-script.

Further developments

JDemetra+ will undergo a major change at the end of 2021 with the release of version 3.0, embedding algorithms for high frequency data among other innovations. The core packages of RJDemetra and the cruncher will be heavily upgraded in the process. However, the versions presented in this paper will keep functioning with the version 2 of JD+, which will be maintained for at least 3 years.

References

- [1] Jean Palate Alain Quartier-la-Tente Anna Michalek and Raf Baeyens. *RJDemetra: Interface to 'JDemetra+' Seasonal Adjustment Software*. R package version 0.1.6. URL: <https://CRAN.R-project.org/package=RJDemetra>. (2018).
- [2] Sylwia Grudkowska et al. *Online JD+ documentation*. URL: <https://jdemetradocumentation.github.io/JDemetra-documentation>.
- [3] Alain Quartier-la-Tente. *rjdworkspace: manipulation of JDemetra+ workspaces*. URL: <https://github.com/InseeFrLab/rjdworkspace>. R package version 0.1.0.
- [4] Alain Quartier-la-Tente. *rjdqa: quality assessment for seasonal adjustment*. URL: <https://CRAN.R-project.org/package=rjdqa>. R package version 0.1.1.
- [5] Alain Quartier-la-Tente. *rjdmarkdown: an 'rmarkdown' extension for formatted 'RJDemetra' outputs*. URL: <https://CRAN.R-project.org/package=rjdmarkdown>. R package version 0.1.0.
- [6] Alain Quartier-la-Tente. *ggdemetra: a ggplot2 extension for seasonal and trading day adjustment with RJDemetra*. URL: <https://CRAN.R-project.org/package=ggdemetra/>. R package version 0.2.2.
- [7] Gianluca Caporello and Agustin Maravall. *Program TSW Revised Reference Manual*. 2004. URL: <https://www.bde.es/f/webbde/SES/Secciones/Publicaciones/PublicacionesSeriadas/DocumentosOcasionales/04/Fic/do0408e.pdf>. Banco de España.
- [8] The US Census Bureau. *The X-13ARIMA-SEATS seasonal adjustment program*. URL: <https://www.census.gov/srd/www/x13as>.

Série des Documents de Travail « Méthodologie Statistique »

9601 : Une méthode synthétique, robuste et efficace pour réaliser des estimations locales de population.
G. DECAUDIN, J.-C. LABAT

9602 : Estimation de la précision d'un solde dans les enquêtes de conjoncture auprès des entreprises.
N. CARON, P. RAVALET, O. SAUTORY

9603 : La procédure FREQ de SAS - Tests d'indépendance et mesures d'association dans un tableau de contingence.
J. CONFAIS, Y. GRELET, M. LE GUEN

9604 : Les principales techniques de correction de la non-réponse et les modèles associés.
N. CARON

9605 : L'estimation du taux d'évolution des dépenses d'équipement dans l'enquête de conjoncture : analyse et voies d'amélioration.
P. RAVALET

9606 : L'économétrie et l'étude des comportements. Présentation et mise en œuvre de modèles de régression qualitatifs. Les modèles univariés à résidus logistiques ou normaux (LOGIT, PROBIT).
S. LOLLIVIER, M. MARPSAT, D. VERGER

9607 : Enquêtes régionales sur les déplacements des ménages : l'expérience de Rhône-Alpes.
N. CARON, D. LE BLANC

9701 : Une bonne petite enquête vaut-elle mieux qu'un mauvais recensement ?
J.-C. DEVILLE

9702 : Modèles univariés et modèles de durée sur données individuelles.
S. LOLLIVIER

9703 : Comparaison de deux estimateurs par le ratio stratifiés et application

aux enquêtes auprès des entreprises.

N. CARON, J.-C. DEVILLE

9704 : La faisabilité d'une enquête auprès des ménages.
1. au mois d'août.
2. à un rythme hebdomadaire
C. LAGARENNE, C. THIESSET

9705 : Méthodologie de l'enquête sur les déplacements dans l'agglomération toulousaine.
P. GIRARD.

9801 : Les logiciels de désaisonnalisation TRAMO & SEATS : philosophie, principes et mise en œuvre sous SAS.
K. ATTAL-TOUBERT, D. LADIRAY

9802 : Estimation de variance pour des statistiques complexes : technique des résidus et de linéarisation.
J.-C. DEVILLE

9803 : Pour essayer d'en finir avec l'individu Kish.
J.-C. DEVILLE

9804 : Une nouvelle (encore une !) méthode de tirage à probabilités inégales.
J.-C. DEVILLE

9805 : Variance et estimation de variance en cas d'erreurs de mesure non corrélées ou de l'intrusion d'un individu Kish.
J.-C. DEVILLE

9806 : Estimation de précision de données issues d'enquêtes : document méthodologique sur le logiciel POULPE.
N. CARON, J.-C. DEVILLE, O. SAUTORY

9807 : Estimation de données régionales à l'aide de techniques d'analyse multidimensionnelle.
K. ATTAL-TOUBERT, O. SAUTORY

9808 : Matrices de mobilité et calcul de la précision associée.
N. CARON, C. CHAMBAZ

9809 : Échantillonnage et stratification : une étude empirique des gains de précision.
J. LE GUENNEC

9810 : Le Kish : les problèmes de réalisation du tirage et de son extrapolation.
C. BERTHIER, N. CARON, B. NEROS

9901 : Perte de précision liée au tirage d'un ou plusieurs individus Kish.
N. CARON

9902 : Estimation de variance en présence de données imputées : un exemple à partir de l'enquête Panel Européen.
N. CARON

0001 : L'économétrie et l'étude des comportements. Présentation et mise en œuvre de modèles de régression qualitatifs. Les modèles univariés à résidus logistiques ou normaux (LOGIT, PROBIT) (version actualisée).
S. LOLLIVIER, M. MARPSAT, D. VERGER

0002 : Modèles structurels et variables explicatives endogènes.
J.-M. ROBIN

0003 : L'enquête 1997-1998 sur le devenir des personnes sorties du RMI - Une présentation de son déroulement.
D. ENEAU, D. GUILLEMOT

0004 : Plus d'amis, plus proches ? Essai de comparaison de deux enquêtes peu comparables.
O. GODECHOT

0005 : Estimation dans les enquêtes répétées : application à l'Enquête Emploi en Continu.
N. CARON, P. RAVALET

0006 : Non-parametric approach to the cost-of-living index.
F. MAGNIEN, J. POUGNARD

0101 : Diverses macros SAS : Analyse exploratoire des données, Analyse des séries temporelles.
D. LADIRAY

0102 : Économétrie linéaire des panels : une introduction.
T. MAGNAC

0201 : Application des méthodes de calages à l'enquête EAE-Commerce.
N. CARON

C 0201 : Comportement face au risque et à l'avenir et accumulation patrimoniale - Bilan d'une expérimentation.
L. ARRONDEL, A. MASSON, D. VERGER

C 0202 : Enquête Méthodologique Information et Vie Quotidienne - Tome 1 : bilan du test 1, novembre 2002.
J.-A. VALLET, G. BONNET, J.-C. EMIN, J. LEVASSEUR, T. ROCHER, P. VRIGNAUD, X. D'HAULTFOEUILLE, F. MURAT, D. VERGER, P. ZAMORA

0203 : General principles for data editing in business surveys and how to optimise it.
P. RIVIERE

0301 : Les modèles logit polytomiques non ordonnés : théories et applications.
C. AFSA ESSAFI

0401 : Enquête sur le patrimoine des ménages - Synthèse des entretiens monographiques.
V. COHEN, C. DEMMER

0402 : La macro SAS CUBE d'échantillonnage équilibré
S. ROUSSEAU, F. TARDIEU

0501 : Correction de la non-réponse et calage de l'enquête Santé 2002
N. CARON, S. ROUSSEAU

0502 : Correction de la non-réponse par ré pondération et par imputation
N. CARON

0503 : Introduction à la pratique des indices statistiques - notes de cours
J-P BERTHIER

0601 : La difficile mesure des pratiques dans le domaine du sport et de la culture - bilan d'une opération méthodologique
C. LANDRE, D. VERGER

0801 : Rapport du groupe de réflexion sur la qualité des enquêtes auprès des ménages
D. VERGER

M2013/01 : La régression quantile en pratique
P. GIVORD, X. D'HAULTFOEUILLE

M2014/01 : La microsimulation dynamique : principes généraux et exemples en langage R
D. BLANCHET

M2015/01 : la collecte multimode et le paradigme de l'erreur d'enquête totale
T. RAZAFINDROVONA

M2015/02 : Les méthodes de Pseudo-Panel
M. GUILLERM

M2015/03 : Les méthodes d'estimation de la précision pour les enquêtes ménages de l'Insee tirées dans Octopusse
E. GROS, K. MOUSSALAM

M2016/01 : Le modèle Logit Théorie et application.
C. AFSA

M2016/02 : Les méthodes d'estimation de la précision de l'Enquête Emploi en Continu
E. GROS, K. MOUSSALAM

M2016/03 : Exploitation de l'enquête expérimentale Vols, violence et sécurité.
T. RAZAFINDROVONA

M2016/04 : Savoir compter, savoir coder. Bonnes pratiques du statisticien en programmation.
E. L'HOURL, R. LE SAOUT, B. ROUPPERT

M2016/05 : Les modèles multiniveaux
P. GIVORD, M. GUILLERM


M2016/06 : Econométrie spatiale : une introduction pratique
P. GIVORD, R. LE SAOUT


M2016/07 : La gestion de la confidentialité pour les données individuelles
M. BERGEAT

M2016/08 : Exploitation de l'enquête expérimentale Logement internet-papier
T. RAZAFINDROVONA

M2017/01 : Exploitation de l'enquête expérimentale Qualité de vie au travail

T. RAZAFINDROVONA

M2018/01 : Estimation avec le score de propension sous 
S. QUANTIN

M2018/02 : Modèles semi-paramétriques de survie en temps continu sous 
S. QUANTIN

M2019/01 : Les méthodes de décomposition appliquées à l'analyse des inégalités
B. BOUTCHENIK, E. COUDIN, S. MAILLARD

M2020/01 : L'économétrie en grande dimension
J. L'HOURL

M2021/01 : R Tools for JDemetra+ - Seasonal adjustment made easier
A. SMYK, A. TCHANG