RJDemetra: an R interface to JDemetra+

Alain Quartier-la-Tente
Insee, Seasonal Adjustment Centre of Excellence (SACE)
alain.quartier@yahoo.fr
1. Introduction to seasonal adjustment (SA)

2. RJDemetra

3. Around RJDemetra and JDemetra+

4. Installation and future developments
Introduction to SA with a boring example? 😞

Figure 1: Industrial production index in France
Introduction to SA with an \( \text{R} \) example

Figure 2: Monthly CRAN downloads of ggplot2
Introduction to trading days adjustment

<table>
<thead>
<tr>
<th>Weekday</th>
<th>CRAN downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday</td>
<td>40217</td>
</tr>
<tr>
<td>Tuesday</td>
<td>42991</td>
</tr>
<tr>
<td>Wednesday</td>
<td>44177</td>
</tr>
<tr>
<td>Thursday</td>
<td>41672</td>
</tr>
<tr>
<td>Friday</td>
<td>35544</td>
</tr>
<tr>
<td>Saturday</td>
<td>15481</td>
</tr>
<tr>
<td>Sunday</td>
<td>16081</td>
</tr>
</tbody>
</table>

Table 1: Total CRAN downloads of officer per weekday since 2017-03
Why and how perform seasonal adjustment?

Purpose of seasonal adjustment: analyse the CRAN downloads of your package but also . . .

- Time comparison (outlook, short-term evolution . . .)
- Spatial comparison
Why and how perform seasonal adjustment?

Purpose of seasonal adjustment: analyse the CRAN downloads of your package but also...

- Time comparison (outlook, short-term evolution...) 
- Spatial comparison

Two leading methods:

- TRAMO/SEATS+ (Bank of Spain)
- X-12ARIMA/X-13ARIMA-SEATS (US-Census Bureau).
Why and how perform seasonal adjustment?

Purpose of seasonal adjustment: analyse the CRAN downloads of your package but also...

- Time comparison (outlook, short-term evolution...)
- Spatial comparison

Two leading methods:

- TRAMO/SEATS+ (Bank of Spain)
- X-12ARIMA/X-13ARIMA-SEATS (US-Census Bureau).

→ proceed in two steps:

1. Pre-adjusting the series of deterministics effects with a RegARIMA model
2. Decomposition: to extract seasonal component
What’s JDemetra+?

TRAMO/SEATS+ and X-13ARIMA-SEATS are implemented in JDemetra+ (JD+)

👍 Software officially recommended by Eurostat and the ECB for seasonal and calendar adjustment of official statistics

→ RJDemetra is an R interface to JDemetra+ based on the Java libraries of JD+
Sommaire

1. Introduction to seasonal adjustment (SA)

2. RJDemetra
   2.1 Current status
   2.2 RegARIMA examples
   2.3 Seasonal adjustment examples
   2.4 Export a JD+ workspace
   2.5 Import a JD+ workspace
   2.6 Reduce time computation

3. Around RJDemetra and JDemetra+

4. Installation and future developments
Current status

- RegARIMA, TRAMO-SEATS and X-13-ARIMA:
  - pre-defined and user-defined specifications: outliers detection, ARIMA detection, user-defined regressors, transformation function...
  - S3 classes with plot, summary, print methods

- Manipulate JD+ workspaces:
  - Import JD+ workspace to get input raw series or SA model
  - Export R models created via RJDemetra

- Include a dataset: industrial production indices in manufacturing in the European Union
Object structure

A SA object is a list() of 5 elements:

```
SA
- regarima (# X-13 and TRAMO-SEAT)
  - specification
  - ...
- decomposition (# X-13 and TRAMO-SEAT)
  - specification
  - ...
- final
  - series
  - forecasts
- diagnostics
  - variance_decomposition
  - combined_test
  - ...
- user_defined
```
Create your first model

Like in JD+ users can defined their own specification or use a pre-defined one:

```r
library(RJDemetra)
ipi_fr <- ipi_c_eu[, "FR"]
ts_mod <- tramoseats(ipi_fr, spec = "RSAfull")
x13_usr_spec <- x13_spec(spec = c("RSA5c"),
                  usrdef.outliersEnabled = TRUE,
                  usrdef.outliersType = c("LS", "AO"),
                  usrdef.outliersDate = c("2008-10-01", "2002-01-01"),
                  usrdef.outliersCoef = c(36, 14),
                  transform.function = "None")
x13_mod <- x13(ipi_fr, x13_usr_spec, userdefined = "diagnostics.ic-ratio")
```

Use `user_defined_variables()` to get the names of the user-defined variables
RegARIMA examples (1/2)

summary(x13_mod$regarima)

## y = regression model + arima (0, 1, 1, 0, 1, 1)
##
## Model: RegARIMA - X13
## Estimation span: from 1-1990 to 12-2017
## Log-transformation: no
## Regression model: no mean, no trading days effect, no leap year effect, Easter
## Regression model: no mean, no trading days effect, no leap year effect, Easter
##
## Coefficients:
## ARIMA:
## Estimate Std. Error T-stat Pr(>|t|)
## Theta(1) -0.53675 0.04770 -11.25 <2e-16 ***
## BTheta(1) -0.50830 0.04961 -10.25 <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Coefficients
## Regression model:
## Estimate Std. Error T-stat Pr(>|t|)
## Easter [1] -1.1686 0.3385 -3.452 0.000629 ***
## A0 (9-2008) 31.4099 2.1812 14.400 < 2e-16 ***
## LS (9-2008) -56.6477 2.2561 -25.109 < 2e-16 ***
## TC (9-2008) 24.1814 3.2563 7.426 1.00e-12 ***
## LS (12-2001) -14.6482 1.6811 -8.714 < 2e-16 ***
RegARIMA examples (2/2)

```r
layout(matrix(1:6, 3, 2)); plot(x13_mod$regarima, ask = FALSE)
```

- Residuals plot showing time series from 1995 to 2015.
- ACF of residuals plot.
- Histogram of residuals.
- Normal Q–Q plot.
- Partial ACF of residuals plot.
- Decomposition graph with time series from 1990 to 2015.
Seasonal adjustment examples (1/7): decomposition

```r
x13_mod$decomposition
```

```r
## Monitoring and Quality Assessment Statistics:
## M stats
## M(1) 0.055
## M(2) 0.041
## M(3) 0.926
## M(4) 0.621
## M(5) 0.724
## M(6) 0.215
## M(7) 0.074
## M(8) 0.208
## M(9) 0.056
## M(10) 0.158
## M(11) 0.146
## Q 0.297
## Q-M2 0.329
##
## Final filters:
## Seasonal filter: 3x5
## Trend filter: 13 terms Henderson moving average
```
Seasonal adjustment examples (2/7): decomposition

```
ts_mod$decomposition

## Model
## AR : 1 + 0.352498 B + 0.133616 B^2
## D : 1 - B - B^12 + B^13
## MA : 1 - 0.186819 B - 0.610856 B^12 + 0.114119 B^13
##
## ## SA
## D : 1 - 2.000000 B + B^2
## MA : 1 - 1.314459 B + 0.340427 B^2
## Innovation variance: 0.4669153
##
## ## Trend
## D : 1 - 2.000000 B + B^2
## MA : 1 + 0.040206 B - 0.959794 B^2
## Innovation variance: 0.04869563
##
## ## Seasonal
## AR : 1 + 0.352498 B + 0.133616 B^2
## 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 + 0.717848 B + 0.460721 B^2 + 0.310085 B^3 + 0.132447 B^4 - 0.049053 B^5 - 0.216655 B^6 - 0.354556 B^7 - 0.445030 B^8 - 0.469587 B^9 - 0.376625 B^10 - 0.166397 B^11 - 0.410618 B^12 - 0.132580 B^13
## Innovation variance: 0.1601924
##
## ## Irregular
## Innovation variance: 0.2056884
```

14 / 33
Seasonal adjustment examples (3/7)

```r
plot(x13_mod$decomposition)
```

![Seasonal adjustment example graph](image-url)
## Last observed values

<table>
<thead>
<tr>
<th>Month</th>
<th>y</th>
<th>sa</th>
<th>t</th>
<th>s</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan</td>
<td>97.4</td>
<td>100.6172</td>
<td>100.6174</td>
<td>-3.2172329</td>
<td>-0.0001992082</td>
</tr>
<tr>
<td>Feb</td>
<td>97.5</td>
<td>100.3127</td>
<td>101.0283</td>
<td>-2.8126932</td>
<td>-0.7155966863</td>
</tr>
<tr>
<td>Mar</td>
<td>112.0</td>
<td>102.5469</td>
<td>101.4894</td>
<td>9.4530696</td>
<td>1.0575376567</td>
</tr>
<tr>
<td>Apr</td>
<td>103.0</td>
<td>101.0897</td>
<td>101.9282</td>
<td>1.9103111</td>
<td>-0.8385432983</td>
</tr>
<tr>
<td>May</td>
<td>100.4</td>
<td>103.0319</td>
<td>102.3136</td>
<td>-2.6318733</td>
<td>0.7182480125</td>
</tr>
<tr>
<td>Jun</td>
<td>111.2</td>
<td>102.4926</td>
<td>102.6921</td>
<td>8.7074293</td>
<td>-0.1994894034</td>
</tr>
<tr>
<td>Jul</td>
<td>103.4</td>
<td>103.1596</td>
<td>103.0816</td>
<td>0.2404277</td>
<td>0.0779236963</td>
</tr>
<tr>
<td>Aug</td>
<td>79.3</td>
<td>103.2483</td>
<td>103.5055</td>
<td>-23.9483256</td>
<td>-0.2572170473</td>
</tr>
<tr>
<td>Sep</td>
<td>109.7</td>
<td>103.5536</td>
<td>103.9555</td>
<td>6.1464361</td>
<td>-0.4019376040</td>
</tr>
<tr>
<td>Oct</td>
<td>114.0</td>
<td>106.6886</td>
<td>104.3955</td>
<td>7.3113786</td>
<td>2.2931579296</td>
</tr>
<tr>
<td>Nov</td>
<td>107.7</td>
<td>105.4631</td>
<td>104.7505</td>
<td>2.2369236</td>
<td>0.7125546908</td>
</tr>
<tr>
<td>Dec</td>
<td>101.4</td>
<td>104.7490</td>
<td>105.0214</td>
<td>-3.3490189</td>
<td>-0.2723590878</td>
</tr>
</tbody>
</table>

## Forecasts:

<table>
<thead>
<tr>
<th>Month</th>
<th>y_f</th>
<th>sa_f</th>
<th>t_f</th>
<th>s_f</th>
<th>i_f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan</td>
<td>101.96630</td>
<td>105.0963</td>
<td>105.1795</td>
<td>-3.1299775</td>
<td>-0.083200162</td>
</tr>
<tr>
<td>Feb</td>
<td>102.23632</td>
<td>105.1464</td>
<td>105.2838</td>
<td>-2.9100563</td>
<td>-0.137428535</td>
</tr>
<tr>
<td>Mar</td>
<td>113.85794</td>
<td>105.5026</td>
<td>105.3966</td>
<td>8.3553336</td>
<td>0.105971540</td>
</tr>
<tr>
<td>Apr</td>
<td>108.47477</td>
<td>105.4896</td>
<td>105.5573</td>
<td>2.9851827</td>
<td>-0.067754048</td>
</tr>
<tr>
<td>May</td>
<td>103.22164</td>
<td>105.7963</td>
<td>105.7844</td>
<td>-2.5746309</td>
<td>0.011859024</td>
</tr>
<tr>
<td>Jun</td>
<td>114.64042</td>
<td>106.0073</td>
<td>106.0629</td>
<td>8.6331483</td>
<td>-0.055612674</td>
</tr>
</tbody>
</table>
Seasonal adjustment examples (5/7)

```r
plot(x13_mod$final, first_date = 2012, type_chart = "sa-trend")
```
Seasonal adjustment examples (6/7)

plot(x13_mod$final, last_date = 2000, type_chart = "cal-seas-irr")

![Seasonal adjustment examples chart](chart.png)
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).

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle</td>
<td>1.557</td>
</tr>
<tr>
<td>Seasonal</td>
<td>39.219</td>
</tr>
<tr>
<td>Irregular</td>
<td>0.362</td>
</tr>
<tr>
<td>TD &amp; Hol.</td>
<td>0.018</td>
</tr>
<tr>
<td>Others</td>
<td>61.971</td>
</tr>
<tr>
<td>Total</td>
<td>103.128</td>
</tr>
</tbody>
</table>

Combined test in the entire series.
Non parametric tests for stable seasonality.

<table>
<thead>
<tr>
<th>Test</th>
<th>P.value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kruskall-Wallis test</td>
<td>0.000</td>
</tr>
<tr>
<td>Test for the presence of seasonality assuming stability</td>
<td>0.000</td>
</tr>
<tr>
<td>Evolutive seasonality test</td>
<td>0.032</td>
</tr>
</tbody>
</table>

Identifiable seasonality present.
Residual seasonality tests.

<table>
<thead>
<tr>
<th>Test</th>
<th>P.value</th>
</tr>
</thead>
<tbody>
<tr>
<td>f-test on sa (seasonal dummies)</td>
<td>0.997</td>
</tr>
<tr>
<td>f-test on i (seasonal dummies)</td>
<td>0.965</td>
</tr>
<tr>
<td>Residual seasonality (entire series)</td>
<td>0.993</td>
</tr>
<tr>
<td>Residual seasonality (last 3 years)</td>
<td>0.922</td>
</tr>
<tr>
<td>f-test on sa (td)</td>
<td>0.001</td>
</tr>
<tr>
<td>f-test on i (td)</td>
<td>0.006</td>
</tr>
</tbody>
</table>
Export a workspace

wk <- new_workspace()
new_multiprocessing(wk, name = "MP-1")
add_sa_item(wk, multiprocessing = "MP-1",
            sa_obj = x13_mod, name = "SA with X13 model 1 ")
add_sa_item(wk, multiprocessing = "MP-1",
            sa_obj = ts_mod, name = "SA with TramoSeats model 1")
save_workspace(wk, "workspace.xml")
Import a workspace

\[
\begin{align*}
\text{wk} & \leftarrow \text{loadworkspace("workspace.xml")} \\
\text{compute}(\text{wk}) & \quad \text{# Important to get the Sa model} \\
\text{models} & \leftarrow \text{get_model(\text{wk}, progress_bar = FALSE)} \quad \text{# get all models} \\
& \quad \text{# Or to get one specific model:} \\
\text{mp} & \leftarrow \text{get_object(\text{wk}, 1)} \\
\text{count}(\text{mp}) & \\
& \quad \text{## [1] 2} \\
\text{sa2} & \leftarrow \text{get_object(\text{mp}, 2)} \\
\text{get_name}(\text{sa2}) & \quad \text{## [1] "SA with TramoSeats model 1"} \\
\text{mod} & \leftarrow \text{get_model(\text{sa2, wk})}
\end{align*}
\]
Manipulate 🍵 objects (1/2)

Default functions can be time consuming (computation of outputs)... Especially if you only need one specific parameter

→ “Manipulate” java models: jx13, jtramoseats, jregarima, jregarima_x13, jregarima_tramoseats and get_jmodel

```r
jx13_mod <- jx13(ipi_fr, x13_usr_spec)
# To get the available outputs:
tail(get_dictionary(jx13_mod), 2)
## [1] "diagnostics.msr-global" "diagnostics.msr(*)"
# To get an indicator:
get_indicators(jx13_mod, "diagnostics.ic-ratio")
## $/grave.ts1 diagnostics.ic-ratio/grave.ts1
## [1] 4.356533
# To get the previous R output
x13_mod <- jSA2R(jx13_mod)
```

→ The output can be customize by every user/institute
Manipulate ☕ objects (1/2)

Default functions can be time consuming (computation of outputs)... Especially if you only need one specific parameter

→ “Manipulate” java models: jx13, jtramoseats, jregarima, jregarima_x13, jregarima_tramoseats and get_jmodel

```r
jx13_mod <- jx13(ipi_fr, x13_usr_spec)
# To get the available outputs:
tail(get_dictionary(jx13_mod), 2)
```

```
## [1] "diagnostics.msr-global" "diagnostics.msr(*)"
```

```
# To get an indicator:
get_indicators(jx13_mod, "diagnostics.ic-ratio")
```

```
## $/grave.ts1
diagnostics.ic-ratio
## [1] 4.356533
```

```
# To get the previous R output
x13_mod <- jSA2R(jx13_mod)
```

→ The output can be customize by every user/institute
Sommaire

1. Introduction to seasonal adjustment (SA)

2. RJDemetra

3. Around RJDemetra and JDemetra+
  3.1 Around RJDemetra
  3.2 Around JDemetra+

4. Installation and future developments
ONE PACKAGE TO RULE THEM ALL
Examples of current use of RJDemetra

- **ggdemetra**: ggplot2 extension for ‘RJDemetra’
  - [https://github.com/AQLT/rjdqa](https://github.com/AQLT/rjdqa)

- **rjdqa**: package to help quality assessment (dashboard and quality report matrix)
  - [https://github.com/AQLT/rjdqa](https://github.com/AQLT/rjdqa)

- **persephone**: enable easy processing during production of SA series (interactive plots, dashboards…)
  - [https://github.com/statistikat/persephone](https://github.com/statistikat/persephone)

- **rjdmardown**: nice rmarkdown outputs for RJDemetra
  - [https://github.com/AQLT/rjdmardown](https://github.com/AQLT/rjdmardown)

- Carry out studies on SA: Ladiray D., Quartier-la-Tente A., “(In)Stability of Reg-ARIMA Models for Seasonal Adjustment”
rjdqa

plot(rjdqa::sa_dashboard(x13_mod))
Seasonal adjustment of the French industrial production index
Around JDemetra+

- State space framework of JD+:  
  🌐 https://github.com/nbbrd/rjdssf

- Benchmarking and temporal disaggregation:  
  🌐 https://github.com/palatej/rjdbench

- R interface to the JWSACruncher (console tool to refresh the models of a JD+ workspace):  
  🌐 https://github.com/AQLT/rjwsacruncher
Sommaire

1. Introduction to seasonal adjustment (SA)

2. RJDemetra

3. Around RJDemetra and JDemetra+

4. Installation and future developments
   4.1 How to install the package?
   4.2 Why use RJDemetra?
   4.3 Future developments
How to install the package?

The package is available on GitHub: https://github.com/jdemetra/rjdemetra

```r
# Cran release
install.packages("RJDemetra")

# Development version
devtools::install_github("jdemetra/rjdemetra")
```

ℹ️ To install it you need Java8: in case you don’t, see the installation manual
Why use RJDemetra?

- Methods used are recommended by Eurostat
- Performance and integration in production with JDemetra+
- Lots of \texttt{R} developments around RJDemetra
- RJDemetra evolves with JDemetra+: will integrate new developments on SA methods
What’s next?

- documentation: article for the Journal of Statistical Software + cheat sheet
- shiny app to change the specification

With JD+ 3.0.0 (by the end of 2020):

- Function to “refresh” the model
- Compatibility with all frequencies (JD+ daily, weekly, etc.)
Thank you for your attention... 

... And don’t forget your stickers!

Alain Quartier-la-Tente  
✉️ alain.quartier@yahoo.fr  🐦 @AlainQlt

jdemetra/rjdemetra  
🐦 @JdemetraPlus