

USER! 2019



RJDemetra: an R interface to JDemetra+

ALAIN QUARTIER-LA-TENTE

Insee, Seasonal Adjustment Centre of Excellence (SACE)
alain.quartier@yahoo.fr

Sommaire

- 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? 😞

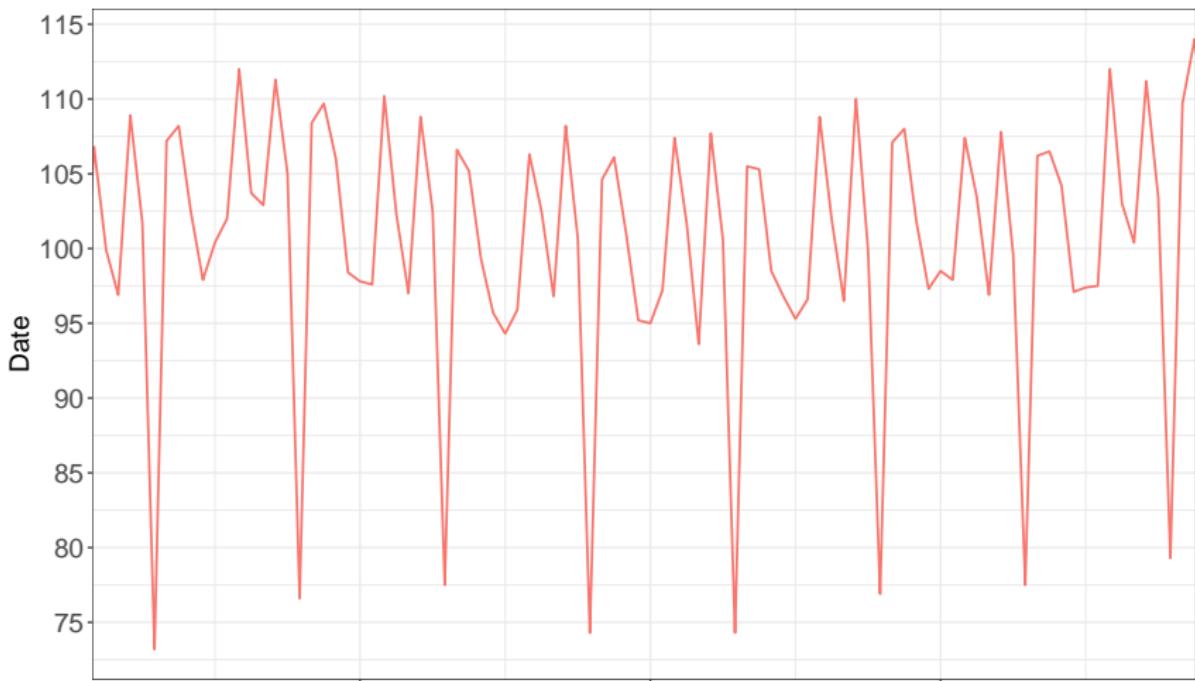


Figure 1: Industrial production index in France

Introduction to SA with an example

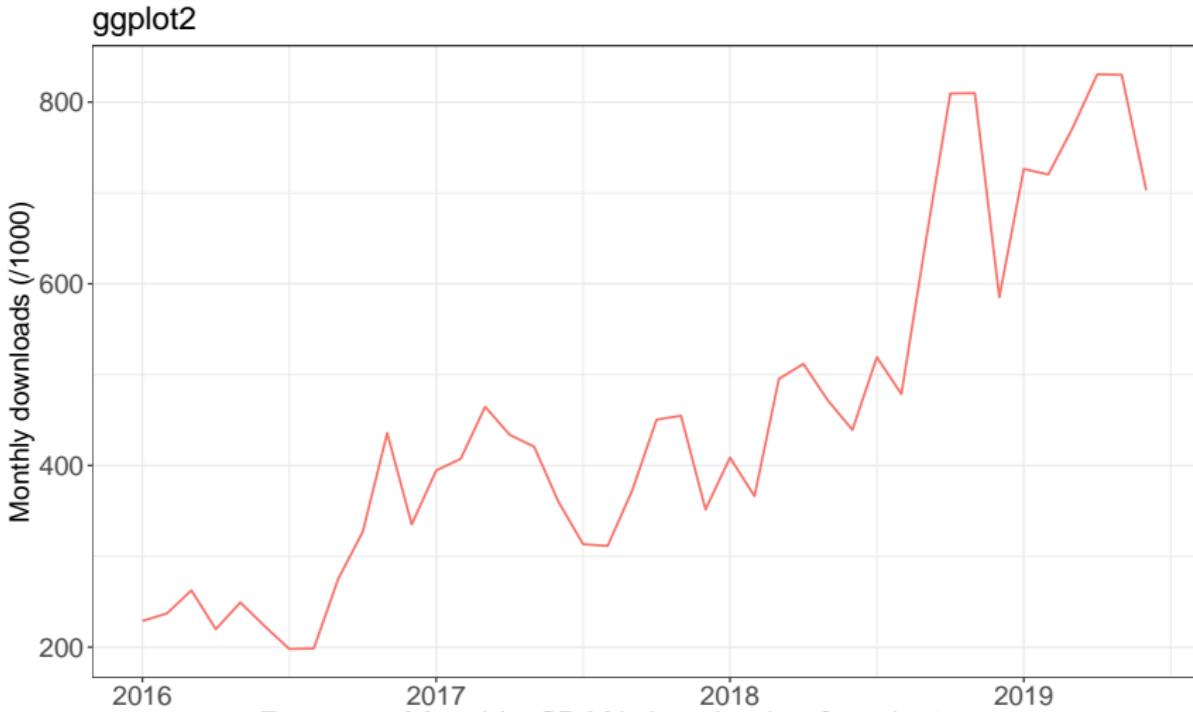


Figure 2: Monthly CRAN downloads of ggplot2

Introduction to trading days adjustment

Weekday	CRAN downloads
Monday	40217
Tuesday	42991
Wednesday	44177
Thursday	41672
Friday	35544
Saturday	15481
Sunday	16081

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+ ?



Time Series Software
for Official Statistics

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 interface to JDemetra+ based on the 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, userdefined 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:

```
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

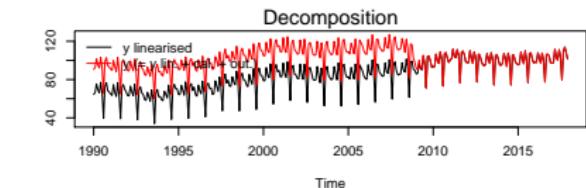
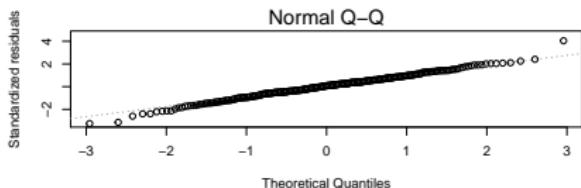
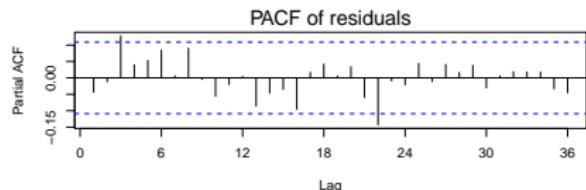
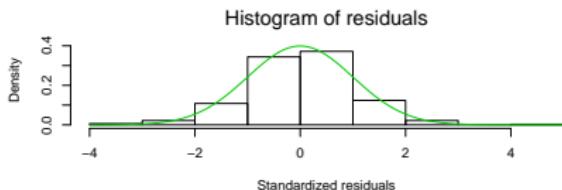
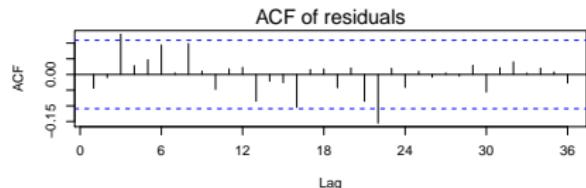
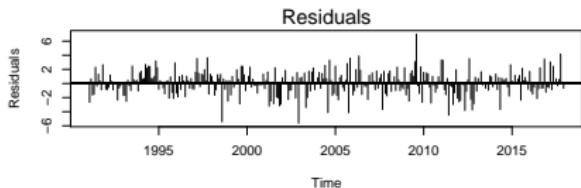
RegARIMA examples (1/2)

```
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
##
## 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
##
## Regression model:
##             Estimate Std. Error  T-stat Pr(>|t|)
## Easter [1]   -1.1686    0.3385  -3.452 0.000629 ***
## AO (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 ***
```

RegARIMA examples (2/2)

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



Seasonal adjustment examples (1/7): decomposition

```
x13_mod$decomposition
```

```
## 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
```

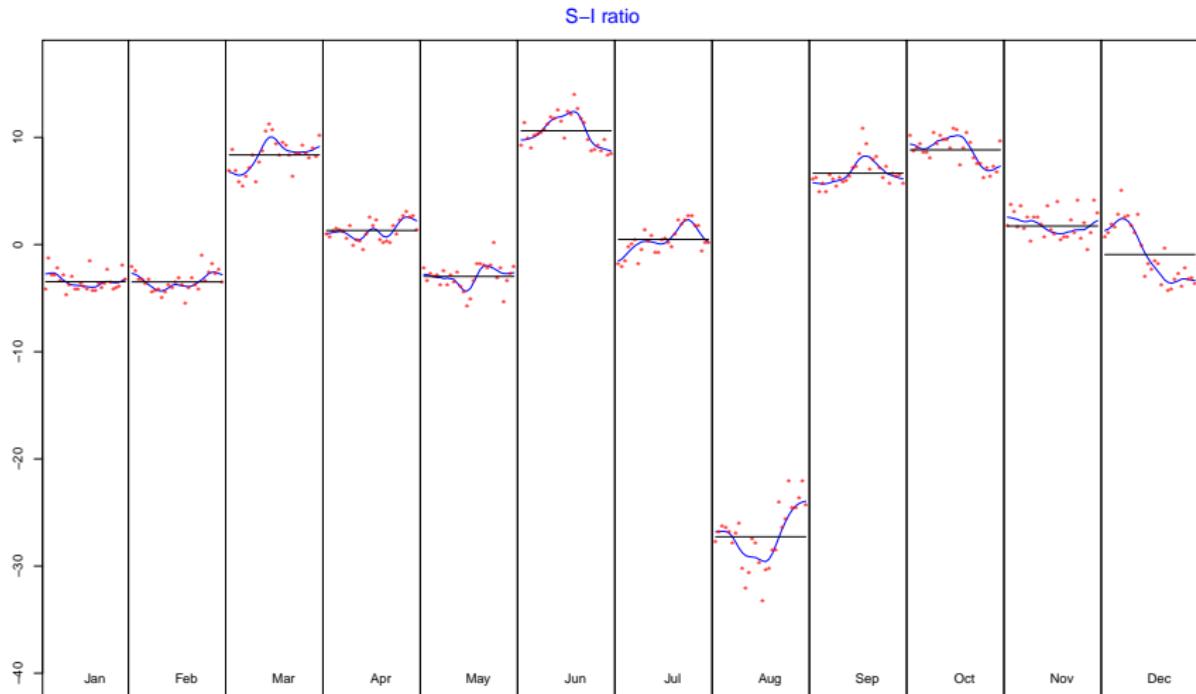
Seasonal adjustment examples (2/7): decomposition

```
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
## Innovation variance: 0.1601924
```

Seasonal adjustment examples (3/7)

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



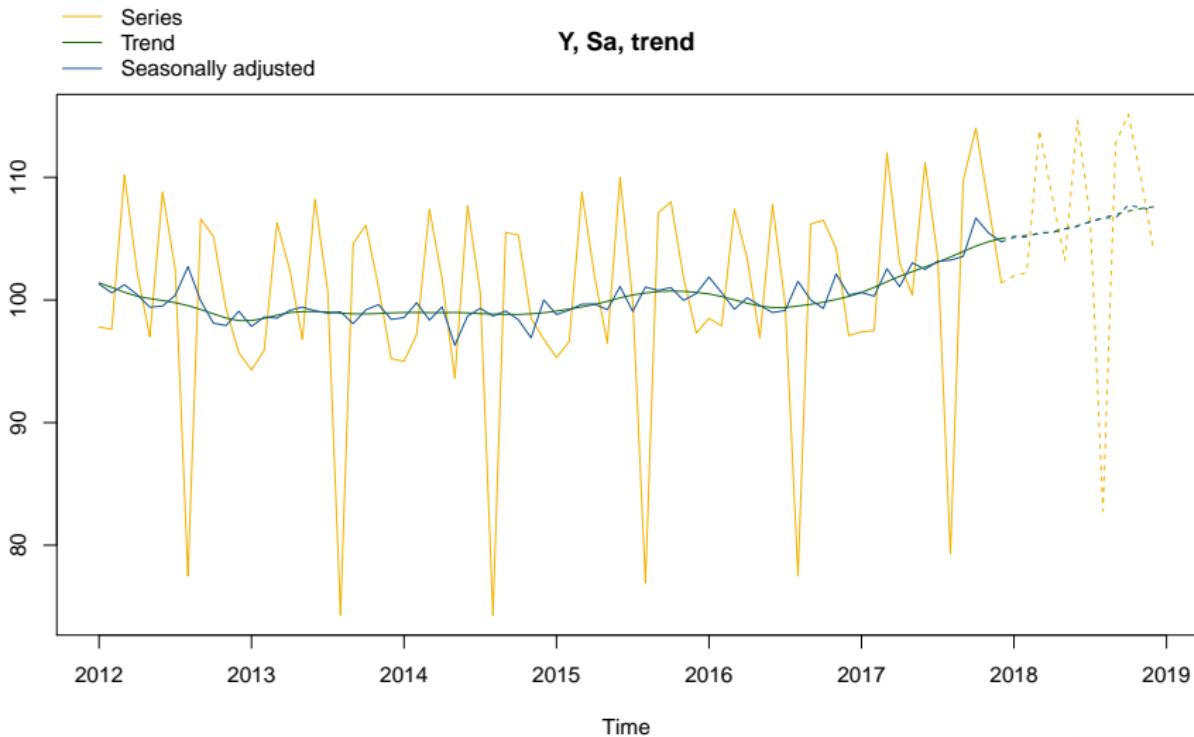
Seasonal adjustment examples (4/7)

x13_mod\$final

```
## Last observed values
##          y      sa      t      s      i
## Jan 2017 97.4 100.6172 100.6174 -3.2172329 -0.0001992082
## Feb 2017 97.5 100.3127 101.0283 -2.8126932 -0.7155966863
## Mar 2017 112.0 102.5469 101.4894  9.4530696  1.0575376567
## Apr 2017 103.0 101.0897 101.9282  1.9103111 -0.8385432983
## May 2017 100.4 103.0319 102.3136 -2.6318733  0.7182480125
## Jun 2017 111.2 102.4926 102.6921  8.7074293 -0.1994894034
## Jul 2017 103.4 103.1596 103.0816  0.2404277  0.0779236963
## Aug 2017  79.3 103.2483 103.5055 -23.9483256 -0.2572170473
## Sep 2017 109.7 103.5536 103.9555  6.1464361 -0.4019376040
## Oct 2017 114.0 106.6886 104.3955  7.3113786  2.2931579296
## Nov 2017 107.7 105.4631 104.7505  2.2369236  0.7125546908
## Dec 2017 101.4 104.7490 105.0214 -3.3490189 -0.2723590878
##
## Forecasts:
##          y_f      sa_f      t_f      s_f      i_f
## Jan 2018 101.96630 105.0963 105.1795 -3.1299775 -0.083200162
## Feb 2018 102.23632 105.1464 105.2838 -2.9100563 -0.137428535
## Mar 2018 113.85794 105.5026 105.3966  8.3553336  0.105971540
## Apr 2018 108.47477 105.4896 105.5573  2.9851827 -0.067754048
```

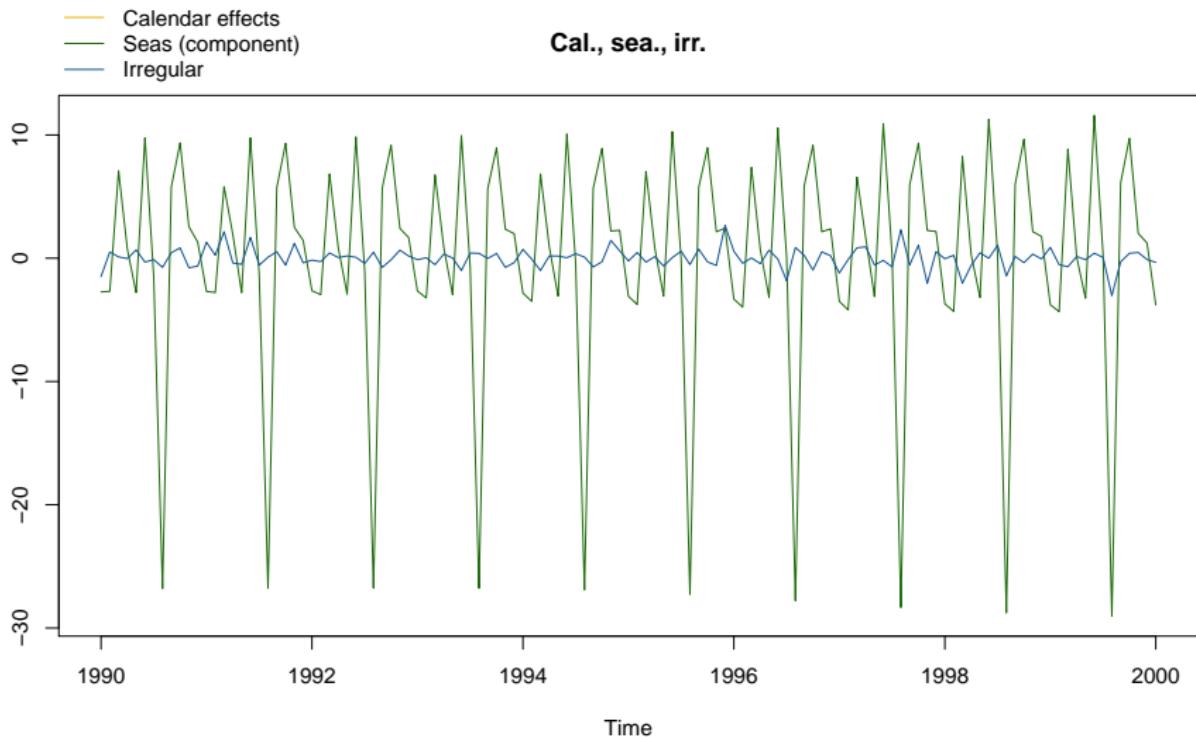
Seasonal adjustment examples (5/7)

```
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 (7/7)

x13_mod\$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      1.557
## Seasonal   39.219
## Irregular   0.362
## TD & Hol.   0.018
## Others     61.971
## Total      103.128
##
## 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    0.032
##
## Identifiable seasonality present
##
```

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")
```

The screenshot shows the RJDemetra application interface. On the left is a tree view of the workspace structure:

- workspace
- Modelling
 - Seasonal adjustment
 - specifications
 - documents
 - multi-documents
 - MP-1
- Utilities
 - Calendars
 - Variables

The main area is titled "MP-1". It contains a processing table:

Series	Method	Estimation	Status	Priority	Quality	Warnings	Comments
SA with X13 model 1	X13		Valid		Good		
SA with TramoSeats model 1	TS		Valid		Severe		

Below the table, there is a detailed view of the "Main results" section for the "SA with X13 model 1" series:

- Input**
- Main results**
- Pre-processing**
- Decomposition (X11)**
- Benchmarking**
- Diagnostics**

SA with X13 model 1

Pre-processing (ReqArima)

Summary

Estimation span: [1-1990 - 12-2017]
336 observations
No trading days effects
No easter effect
7 detected outliers
2 fixed outliers

Import a workspace

```
wk <- load_workspace("workspace.xml")
compute(wk) # Important to get the Sa model
models <- get_model(wk, progress_bar = FALSE) # get all models
# Or to get one specific model:
mp <- get_object(wk, 1)
count(mp)

## [1] 2
sa2 <- get_object(mp, 2)
get_name(sa2)

## [1] "SA with TramoSeats model 1"
mod <- get_model(sa2, wk)
```

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

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

```
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")
```

```
## $`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

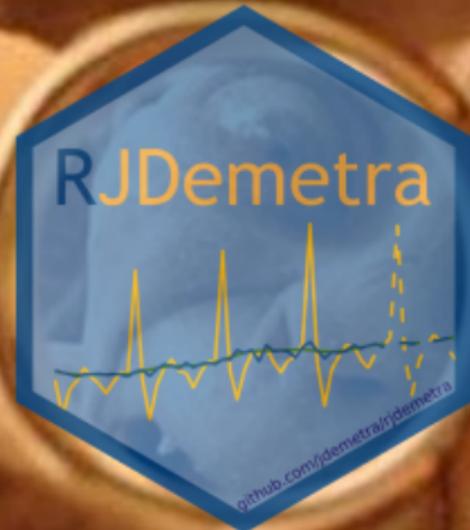
3. Around RJDemetra and JDemetra+

3.1 Around RJDemetra

3.2 Around JDemetra+

4. Installation and future developments

ONE PACKAGE TO RULE THEM ALL

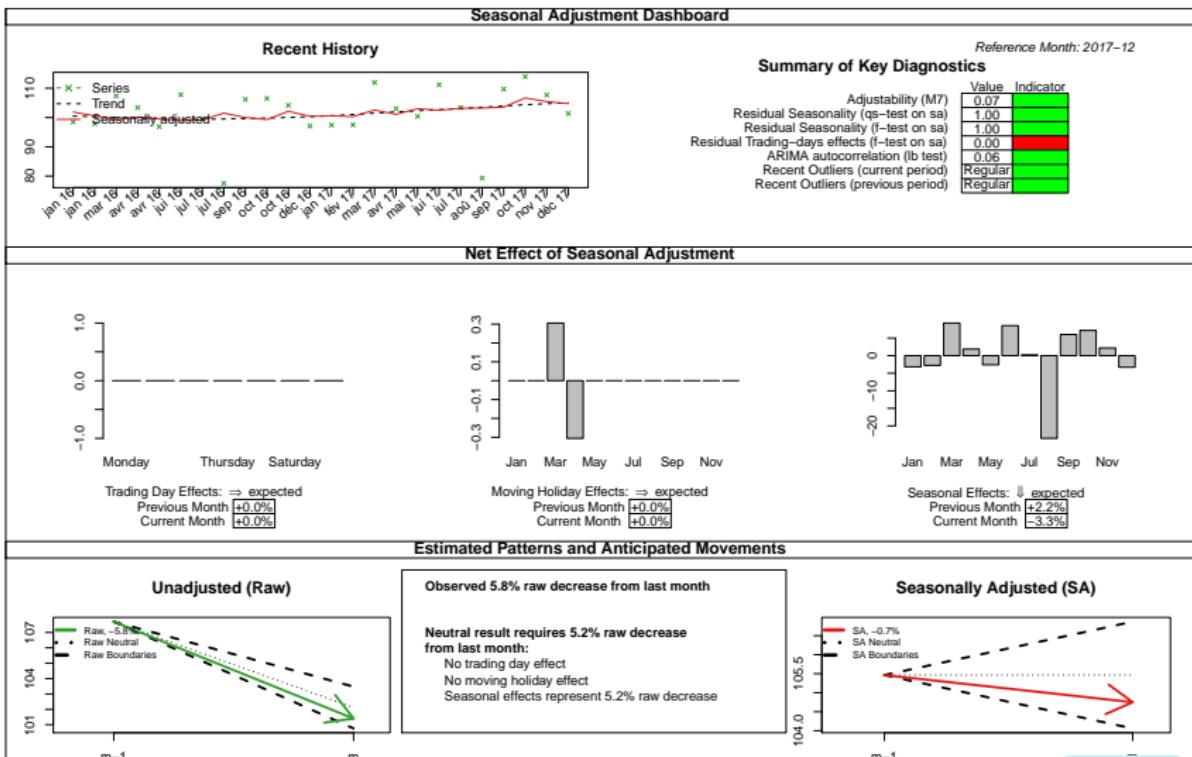


Examples of current use of RJDemetra

- `ggdemetra`: ggplot2 extension for 'RJDemetra'
- ⌚ <https://github.com/AQLT/rjdqa>
- `rjdqa`: package to help quality assessment (dashboard and quality report matrix)
- ⌚ <https://github.com/AQLT/rjdqa>
- `persephone`: enable easy processing during production of SA series (interactive plots, dashboards...)
- ⌚ <https://github.com/statistikat/persephone>
- `rjdmarkdown`: nice rmarkdown outputs for RJDemetra
- ⌚ <https://github.com/AQLT/rjdmarkdown>
- 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))
```





ggdemetra

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 : <https://github.com/jdemetra/rjdemetra>

```
# Cran release  
install.packages("RJDemetra")  
  
# Development version  
devtools::install_github("jdemetra/rjdemetra")
```



To install it you need Java8: in case you don't, see the installation manual
<https://github.com/jdemetra/rjdemetra/wiki/Installation-manual>

Why use RJDemetra ?

- Methods used are recommended by Eurostat
- Performance and integration in production with JDemetra+
- Lots of  developments around RJDemetra
- RJDemetra evolves with JDemetra+: will integrate new developments on SA methods



What's next?

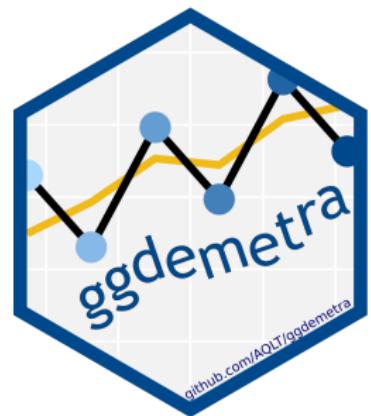
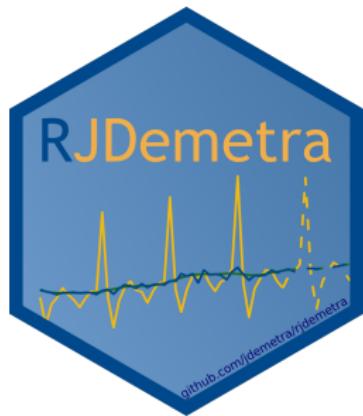
- documentation: article for the Journal of Statistical Software + cheat sheet
- shiny app to change the specification

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](https://github.com/jdemetra/rjdemetra)
 [@JdemetraPlus](https://twitter.com/JdemetraPlus)