NTTS, 13 March 2019

RJDemetra: an R interface to JDemetra+

Alain Quartier-la-Tente and Anna Michalek
Insee, Seasonal Adjustment Centre of Excellence (AQLT) and European Central Bank (AM)

Sommaire

Introduction to seasonal adjustment

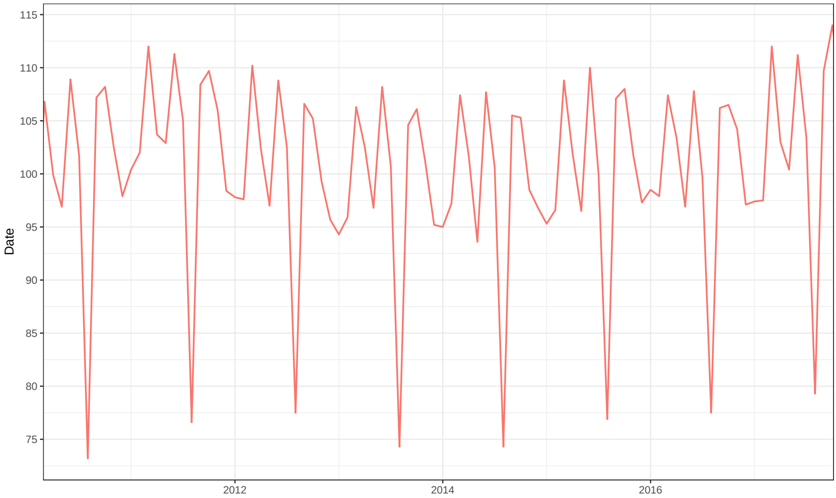


Figure 1: Industrial production index in France

Introduction to seasonal adjustment (2/3)

Purpose of seasonal adjustment:

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

Introduction to seasonal adjustment (2/3)

Purpose of seasonal adjustment:

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

Two leading methods:

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

Introduction to seasonal adjustment (2/3)

Purpose of seasonal adjustment:

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

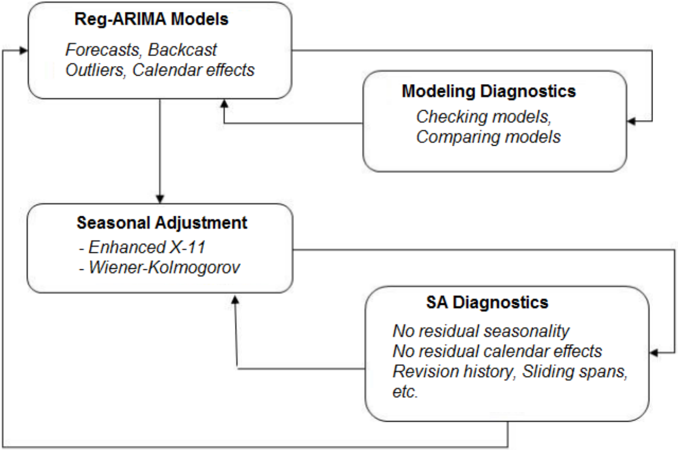
Two leading methods:

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

→ proceed in two steps

Introduction to seasonal adjustment (3/3)

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 R interface to JDemetra+ based on the Java libraries of JD+

Sommaire

Current status

- RegARIMA, TRAMO-SEATS and X-13-ARIMA:
  - pre-defined and user-defined specifications
  - 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

RegARIMA examples (1/3)

```r
library(RJDemetra)
ipi_fr <- ipi_c_eu[,"FR"]
regarima_model <- regarima_def_x13(ipi_fr, spec = "RG4c")
regarima_model
```

```
## y = regression model + arima (2, 1, 1, 0, 1, 1)
## Log-transformation: no
## Coefficients:
##             Estimate Std. Error
## Phi(1)        0.3358      0.171
## Phi(2)        0.2060      0.096
## Theta(1)     -0.2450      0.173
## BTheta(1)    -0.5112      0.050
##
##                 Estimate Std. Error
## Easter [1]        -1.133      0.337
## LS (11-2008)      -8.000      1.283
## LS (1-2009)       -7.551      1.283
```

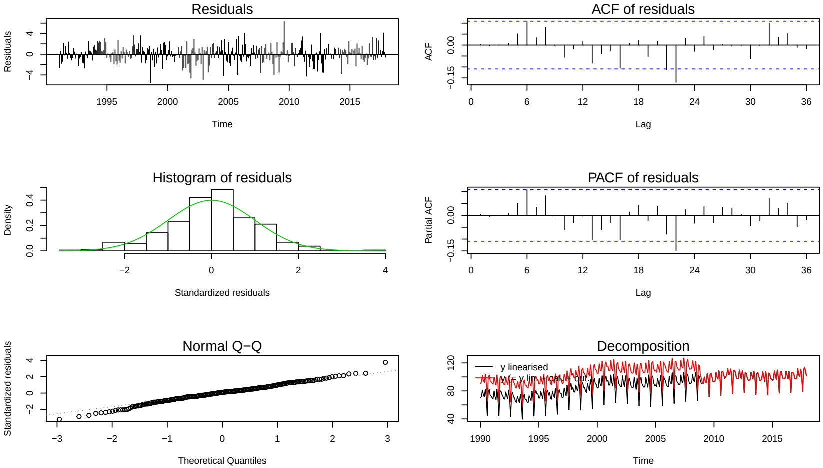
RegARIMA examples (2/3)

```
summary(regarima_model)
```

```
## y = regression model + arima (2, 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, Easte
##
## Coefficients:
## ARIMA:
##           Estimate Std. Error  T-stat Pr(>|t|)
## Phi(1)     0.33579    0.17106   1.963   0.0505 .
## Phi(2)     0.20600    0.09643   2.136   0.0334 *
## Theta(1)  -0.24498    0.17272  -1.418   0.1571
## BTheta(1) -0.51123    0.05004 -10.216   <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]    -1.1332     0.3373 -3.359 0.000875 ***
## LS (11-2008)  -7.9997     1.2831 -6.235 1.42e-09 ***
```

RegARIMA examples (3/3)

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



Seasonal adjustment examples (1/8)

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
```

Seasonal adjustment examples (2/8)

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

```
x13_usr_spec <- x13_spec_def(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)
ts_mod <- tramoseats_def(ipi_fr, spec = "RSAfull")
```

Seasonal adjustment examples (3/8): decomposition

```r
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
```

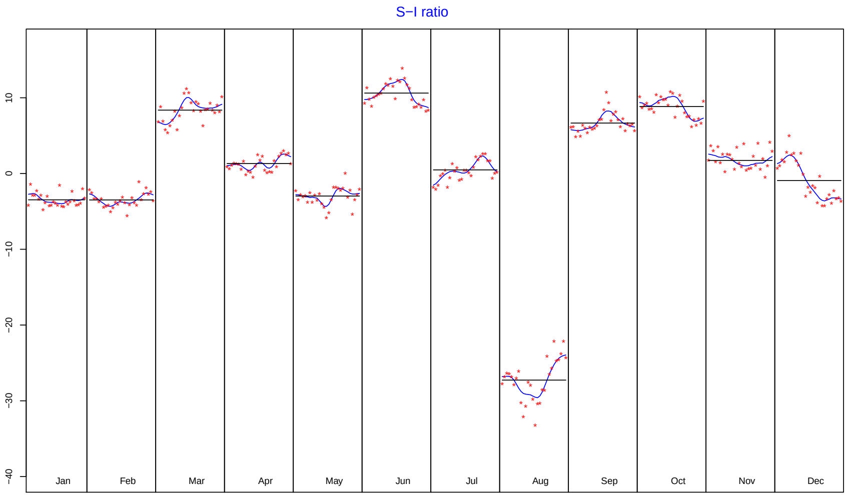
Seasonal adjustment examples (4/8): decomposition

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

Seasonal adjustment examples (5/8)

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



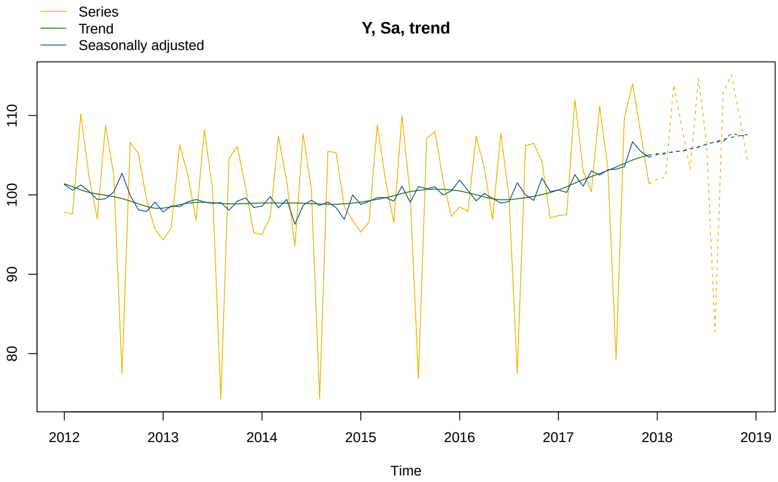
Seasonal adjustment examples (6/8)

```
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 (7/8)

```
plot(x13_mod$final, first_date = 2012, type_chart = "sa-trend")
```



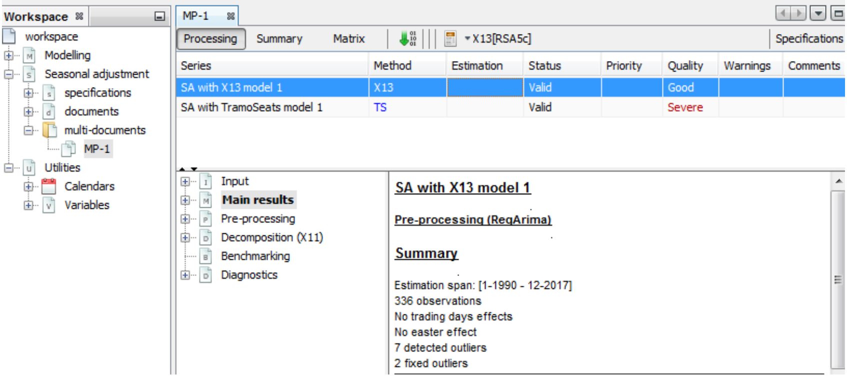
Seasonal adjustment examples (8/8)

```
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

```r
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 (1/3)

```
wk <- load_workspace("workspace.xml")
get_ts(wk)
```

```
## $`MP-1`
## $`MP-1`$`SA with X13 model 1 `
##         Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov
## 1990   90.5  92.6 101.9  95.2  92.1 103.3  91.8  65.5  99.0 102.8  94.3
## 1991   90.9  89.6  99.9  93.3  88.3 103.0  89.7  65.1  98.2 100.8  95.8
## 1992   89.4  89.0  99.5  93.0  89.1 101.3  89.4  64.1  94.9  98.6  92.2
## 1993   85.3  84.3  93.2  87.8  83.5  95.4  86.2  60.1  92.1  95.8  88.1
## 1994   84.9  84.0  94.1  90.1  86.8 100.4  90.8  64.5  96.8 101.0  96.6
## 1995   90.4  90.5 100.4  94.5  89.7 103.7  93.8  65.5  99.7 101.8  94.6
## 1996   90.3  88.8 100.7  93.8  91.2 104.4  92.3  67.2 100.2 102.3  96.9
## 1997   90.5  91.6 104.0  99.7  93.9 108.8  98.2  73.4 105.8 111.8 102.4
## 1998   99.2  99.0 109.4 103.0 100.7 114.8 104.9  73.3 109.6 112.7 105.9
## 1999  100.5  98.6 111.8 104.3 101.3 117.4 106.6  74.9 113.4 118.2 110.9
## 2000  104.8 104.9 118.9 110.2 108.0 122.5 111.8  80.5 117.5 121.7 114.3
## 2001  108.8 109.2 123.7 111.8 108.4 124.7 111.1  84.2 117.8 121.0 111.6
## 2002  106.6 107.0 121.4 112.8 106.4 122.2 109.7  82.3 117.1 118.7 113.0
## 2003  105.4 105.7 120.1 111.1 102.8 118.3 108.8  78.7 115.9 119.9 110.8
## 2004  105.8 107.0 120.0 112.1 105.8 123.6 112.0  78.4 120.0 122.0 112.0
## 2005  109.1 106.7 117.9 113.5 106.8 122.3 110.3  80.0 121.4 118.4 115.2
## 2006  107.3 106.3 121.9 112.5 110.8 126.7 112.5  82.5 122.2 121.9 113.7
```

Import a workspace (2/3)

```
compute(wk) # Important to get the Sa model
models <- get_model(wk) # A progress bar is printed by default
```

```
## Multiprocessing 1 on 1:
##
  |
  |                                                                 |   0%
  |
  |================================                                 |  50%
  |
  |=================================================================| 100%
```

```
# To extract only one model
mp <- get_object(wk, 1)
count(mp)
```

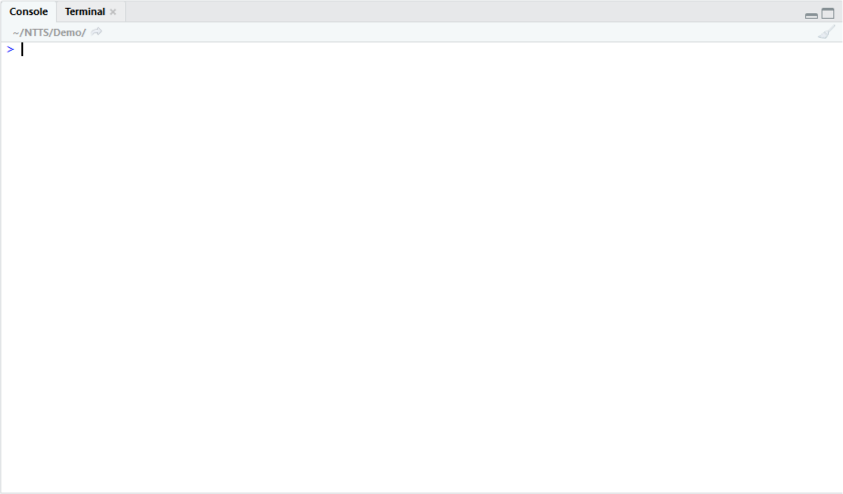
```
## [1] 2
```

```
sa2 <- get_object(mp,2)
get_name(sa2)
```

```
## [1] "SA with TramoSeats model 1"
```

```
mod <- get_model(wk, sa2)
```

Import a workspace (2/3)



How to install the package?

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

It has also it's own website: https://jdemetra.github.io/rjdemetra/

It package can be installed from CRAN:

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

Or from github (development version):

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

To install it you need Java8: in case you don't, install a portable version of Java8 and set the JAVA_HOME path.

What's next? (1/2)

Documentation:

- Vignette/article for the Journal of Statistical Software
- Guide to install the package with portable version of Java (when you don't have administrator rights)
- Cheat sheet

What's next? (2/2)

Package:

- Get only the Java object of a SA (to reduce computation/customize the output)
- Possibility to used user-defined calendar regressors (currently: only user-defined regressors)
- Function to "refresh" the model (JD+ 3.0.0)

Sommaire

Examples of current use of RJDemetra

- rjdqa (experimental, no documentation): 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

- Non explore topics: direct vs indirect adjustment (persephone), analyse of revisions, etc.
- Carry out studies on SA: Ladiray D., Quartier-la-Tente A., "(In)Stability of Reg-ARIMA Models for Seasonal Adjustment" → STS05 in room MANS

Thank you for your attention

jdemetra/rjdemetra
@JdemetraPlus
Other works and packages around JD+: nbbrd

Contact:
Alain Quartier-la-Tente
alain.quartier-la-tente@insee.fr
@AQLT
AQLT