

SACE MEETING #5, 4 OCTOBER 2018



R and JDemetra+ : RJDemetra and rjdqa

ALAIN QUARTIER-LA-TENTE

Insee, Seasonal Adjustment Centre of Excellence (SACE)

Sommaire

1. RJDemetra

- 1.1 Purpose and current status
- 1.2 RegARIMA examples
- 1.3 Seasonal adjustment examples
- 1.4 Manipulate workspaces
- 1.5 How to install and contribute to the package?
- 1.6 Future developments

2. One addin example: `rdjqa`

Purpose of the RJDemetra package

- Complete R package for Tramo-Seats and X13
- Users: “pure R” package
 - Part of R routines, automatization
 - Batch processing
 - E.g.: direct vs indirect aggregates adjustment, dashboards
 - Usage of other R functions and packages
- JD+ functionality
 - Modeling and seasonal adjustment
 - Full specification
- Advanced graphical presentation: JD+

Current status

- RegARIMA, TRAMO-SEATS and X-13-ARIMA:
 - R package with documentation
 - S3 classes with plot, summary, print methods
 - Possibility to add user-defined regressors but not user-defined calendar regressors
- Manipulate workspace (only TRAMO-SEATS and X-13-ARIMA):
 - Import JD+ workspace to get: input raw series or SA model
 - Export R models created via RJDemetra

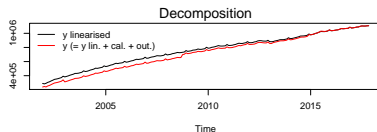
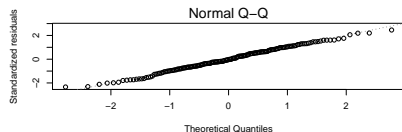
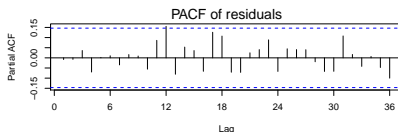
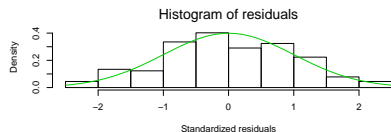
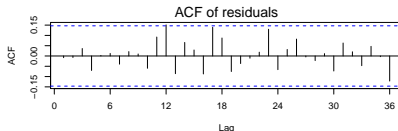
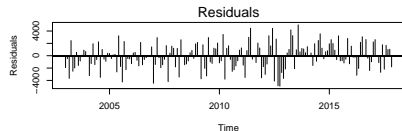
RegARIMA examples (1/3)

```
library(RJDemetra)
regarima_model <- regarima_def_x13(myseries, spec = "RG4c")
regarima_model # Or summary(regarima_model) to have more details
```

```
## y = regression model + arima (1, 1, 2, 0, 1, 1)
## Log-transformation: no
## Coefficients:
##           Estimate Std. Error
## Phi(1)      -0.8317      0.076
## Theta(1)    -0.7989      0.100
## Theta(2)     0.2495      0.081
## BTheta(1)   -0.7631      0.060
##
##           Estimate Std. Error
## Mean          -289.3      129.59
## Week days      -146.6       31.14
## Leap year       1007.0      764.42
## LS (10-2008)   37838.8     1913.00
## LS (1-2003)   -18866.6     2013.42
## AO (1-2002)    14719.0     1536.27
## LS (1-2015)    16158.1     2692.61
## LS (9-2011)     7401.2     1914.06
## AO (4-2012)   -5428.6     1397.40
```

RegARIMA examples (2/3)

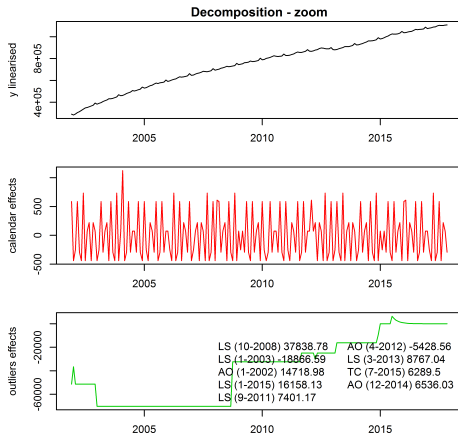
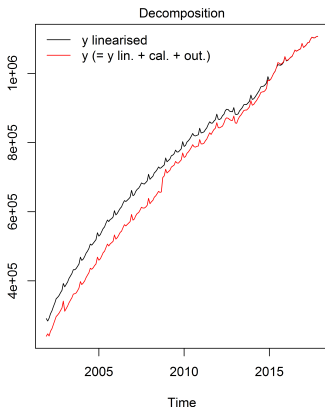
```
layout(matrix(1:6, 3, 2));plot(regarima_model, ask = FALSE)
```



RegARIMA examples (3/3)

To select a specific graph which parameter; `dec_zoom` for an additional regarima decomposition graph:

```
plot(regarima_model, which = 6, dec_zoom = TRUE)
```



Seasonal adjustment examples (1/7)

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

1. `regarima`: the `RegArima` model
2. `decomposition`: decomposition variables (\neq for `TRAMO-SEATS` and `X-13-ARIMA`)
3. `final`: time series main results
4. `diagnostics`: residuals tests, etc.
5. `user_defined`: other user_defined variables not exported by default (see `?user_defined_variables`)

```
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(36000,14000),
                           transform.function = "None")

x13_mod <- x13(myseries, x13_usr_spec)
ts_mod <- tramoseats_def(myseries, spec = "RSAfull")
```


Seasonal adjustment examples (2/7)

```
x13_mod$decomposition
```

```
## Monitoring and Quality Assessment Statistics:
```

```
##           M stats
```

```
## M(1)      0.026
```

```
## M(2)      0.011
```

```
## M(3)      0.000
```

```
## M(4)      0.423
```

```
## M(5)      0.000
```

```
## M(6)      0.095
```

```
## M(7)      0.188
```

```
## M(8)      0.356
```

```
## M(9)      0.128
```

```
## M(10)     0.385
```

```
## Q         0.144
```

```
## Q-M2      0.160
```

```
##
```

```
## Final filters:
```

```
## Seasonal filter: 3x5
```

```
## Trend filter: 9-Henderson
```

Seasonal adjustment examples (3/7)

```
print(ts_mod$decomposition, enable_print_style = FALSE)
```

```
## Model
```

```
## AR : 1 - 0.060534 B - 0.151535 B^2 - 0.346036 B^3 - 0.302623 B^12 + 0.018319
```

```
## D : 1 - B - B^12 + B^13
```

```
## MA : 1 - 0.950000 B^12
```

```
##
```

```
##
```

```
## SA
```

```
## AR : 1 - 0.965728 B - 0.096740 B^2 - 0.208867 B^3 + 0.313230 B^4
```

```
## D : 1 - 2.000000 B + B^2
```

```
## MA : 1 - 1.027671 B - 0.516726 B^2 + 0.212676 B^3 + 0.444341 B^4 + 0.015991
```

```
## Innovation variance: 0.1659023
```

```
##
```

```
## Trend
```

```
## AR : 1 - 1.701590 B + 0.720893 B^2
```

```
## D : 1 - 2.000000 B + B^2
```

```
## MA : 1 - 1.416543 B - 0.489305 B^2 + 1.416934 B^3 - 0.510304 B^4
```

```
## Innovation variance: 0.1113562
```

```
##
```

```
## Seasonal
```

```
## AR : 1 + 0.905194 B + 0.819377 B^2 + 0.741695 B^3 + 0.671378 B^4 + 0.607728
```

```
## D : 1 + B + B^2 + B^3 + B^4 + B^5 + B^6 + B^7 + B^8 + B^9 + B^10 + B^11
```

Seasonal adjustment examples (4/7)

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

S-I ratio



Seasonal adjustment examples (5/7)

```
x13_mod$final
```

```
## Last observed values
```

	y	sa	t	s	i
## Dec 2016	1087082	1075842	1076809	11240.0563	-966.92838
## Jan 2017	1075137	1080586	1080172	-5448.7173	413.95902
## Feb 2017	1078141	1084192	1083675	-6050.8123	516.52696
## Mar 2017	1082422	1086339	1087148	-3917.3376	-808.74611
## Apr 2017	1089204	1090829	1090257	-1625.1053	571.84127
## May 2017	1089662	1092839	1092962	-3177.2306	-122.51812
## Jun 2017	1099145	1096159	1095433	2985.8807	726.30641
## Jul 2017	1104797	1096917	1097978	7880.4569	-1061.84552
## Aug 2017	1102805	1100832	1100754	1972.7310	78.66638
## Sep 2017	1103295	1103768	1103873	-473.3715	-104.16916
## Oct 2017	1106170	1107814	1107030	-1643.7937	783.96096
## Nov 2017	1107118	1108997	1109863	-1879.3688	-865.76589

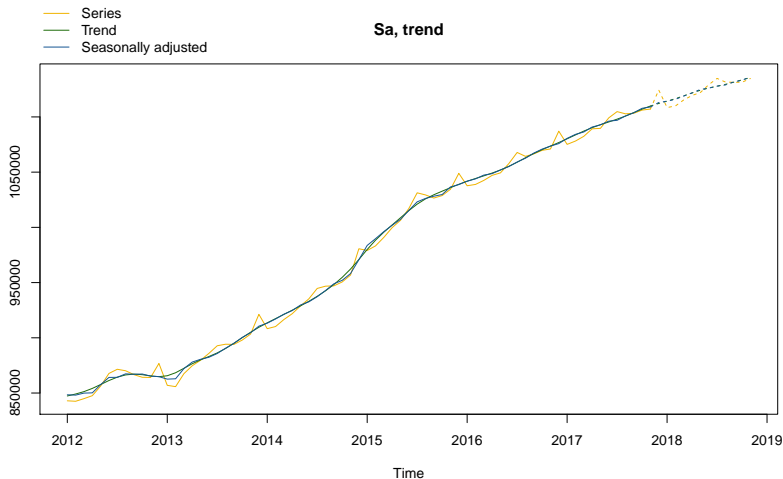
```
##
```

```
## Forecasts:
```

	y_f	sa_f	t_f	s_f	i_f
## Dec 2017	1124358	1113029	1112188	11328.9675	840.81672
## Jan 2018	1108491	1114279	1114294	-5788.7407	-14.26683
## Feb 2018	1109887	1115922	1116527	-6034.7711	-605.30932
## Mar 2018	1115374	1118913	1119053	-3539.3140	-139.75789

Seasonal adjustment examples (6/7)

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



Seasonal adjustment examples (7/7)

```
print(x13_mod$diagnostics, enable_print_style = FALSE)
```

```
## Relative contribution of the components to the stationary portion of the var
## Trend computed by Hodrick-Prescott filter (cycle length = 8.0 years)
##           Component
## Cycle      27.110
## Seasonal    7.795
## Irregular   0.382
## TD & Hol.    0.114
## Others      87.839
## Total      123.240
##
## Residual seasonality tests
##
##                                     P.value
## qs test on sa                      1.000
## qs test on i                       1.000
## f-test on sa (seasonal dummies)    0.967
## f-test on i (seasonal dummies)    0.854
## Residual seasonality (entire series) 0.967
## Residual seasonality (last 3 years) 0.967
## f-test on sa (td)                  0.878
## f-test on i (td)                   0.734
##
```

Export a workspace

```

wk <- new_workspace()
new_multiprocessing(wk, "sa1")
add_sa_item(wk, multiprocessing = "sa1",
            x13_mod)
add_sa_item(wk, multiprocessing = "sa1",
            ts_mod, "TramoSeats")
save_workspace(wk, "workspace.xml")

```

The screenshot shows the RJDemetra software interface. On the left, a tree view displays the workspace structure: 'workspace' contains 'Modelling', 'Seasonal adjustment', 'specifications', 'documents', 'multi-documents', and 'sa1'. 'sa1' contains 'Utilities'. The main panel shows the 'sa1' workspace with a table of series and a list of results.

Series	Method	Estimation	Status	Priority	Quality	Warnings	Com...
x13_mod	X13		Valid		Good		
TramoSeats	TS		Unprocessed				

Below the table, a list of results is shown:

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

The 'Main results' section is expanded, showing:

- [Pre-processing \(RegArima\)](#)
- [Summary](#)

Estimation span: [12-2001 - 11-2017]

Import a workspace (1/3)

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

```
## $sa1
## $sa1$x13_mod
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2001
## 2002  246680  240476  254298  261655  273847  285696  296574  301114
## 2003  312120  319317  327218  336320  343756  351042  361475  362668
## 2004  389117  393469  399591  409373  416623  423014  436246  433400
## 2005  459893  463566  471751  481074  485829  496407  506284  500775
## 2006  520777  524765  532187  540257  543523  553658  562621  558909
## 2007  575640  578720  588522  594796  597610  604978  612926  610644
## 2008  623127  628987  632868  641391  645746  652087  658817  655993
## 2009  712199  715829  719841  729136  731911  734897  745341  741044
## 2010  757084  759517  768585  772608  778963  785493  793930  787969
## 2011  796249  796244  798344  805450  810435  819663  828194  823445
## 2012  842958  842541  844906  847598  856303  867744  871509  870166
## 2013  856975  855786  867536  874708  879698  885908  892804  894196
## 2014  908272  910194  916522  921814  928897  935294  944707  946755
## 2015  979087  983220  990949  999766 1006424 1017083 1031285 1029364
## 2016 1037667 1038867 1042518 1047084 1049349 1057708 1067801 1064305
## 2017 1075137 1078141 1082422 1089204 1089662 1099145 1104797 1102805
```


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] "TramoSeats"
```

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

Import a workspace (3/3)



Still some bugs importing a workspace created by JDemetra+ when:

- The workspace contains user-defined trading days regressors
- The workspace contains an invalid model

How to install the package?

The package is available on github: <https://github.com/nbbird/rjdemetra>



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

To install it use devtools or download the zip file

```
# install.packages("devtools")
devtools::install_github("nbbird/rjdemetra",
                        args = "--no-multiarch")
```

How to contribute to the package?

You can contribute:

- Testing it and reporting issues (<https://github.com/nbbird/rjdemetra/issues>)
- Correcting issues (<https://github.com/nbbird/rjdemetra/pulls>)
- Developing new tools (other packages, new functions, etc.)



What's next?

- Possibility to use user-defined calendar regressors
- update function to refresh a model with new data
- Include a “complete” dataset in the package
- Write a vignette (long-form guide to the package) or an article in the Journal of Statistical Software
- More tests on the package

Sommaire

1. RJDemetra

2. One addin example: rdjqa

2.1 What for?

2.2 Statistics Canada dashboard

What for?



A package for quality assessment for seasonal adjustment. It implements:

- Statistics Canada Dashboard (to provide a snapshot of an individual series at a point in time and points out some possible problems)
- Insee quality report matrix (used to help the analyst during production to prioritize the models to check)

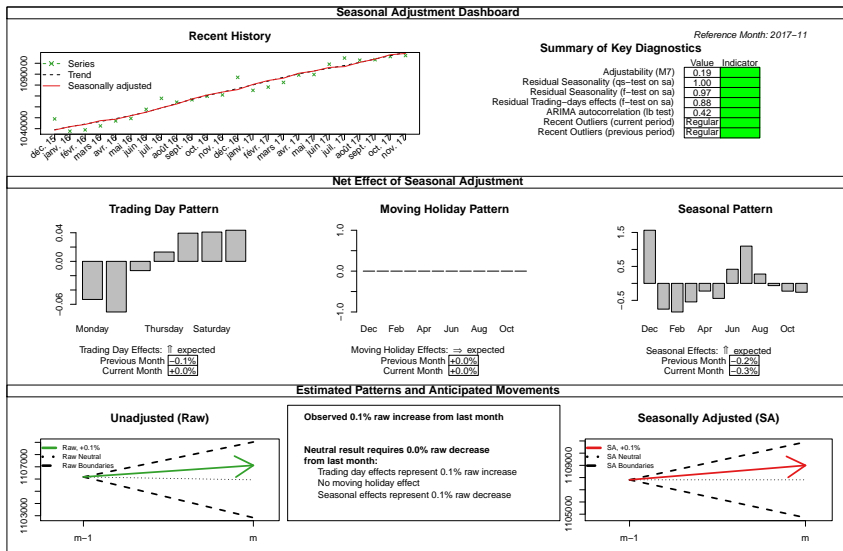
→ See the Seasonal Adjustment handbook

Available on github <https://github.com/AQLT/rjdqa>, still in development (only works for X13 models and no documentation yet)

Example of the dashboard:

```
library(rjdqa)
plot(sa_dashboard(x13_mod))
```

Example of the dashboard (1/2)



Example of the dashboard (2/2)

1. **Recent History of Series:** plot of the raw series, the SA series and the trend for the most recent periods. It is intended to identify trend direction, overall volatility and obvious outliers
2. **Summary of Key Diagnostics:** key diagnostics as residual seasonality, recent and recurring outliers, moving seasonality, ARIMA model autocorrelation
3. **Estimated Patterns and Anticipated Movements:** estimated trading day, moving holiday and seasonal pattern (rescaled in additive decomposition to represent relative level)
4. **Net Effect of Seasonal Adjustment:** movement in the raw series, compared to typical ranges centered around “neutral” value (when $SA_t = SA_{t-1}$)