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

1. RJDemetra

1.1 Purpose and current status

2. Some examples

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

Sommaire

1. RJDemetra

2. Some examples

2.1 RegARIMA examples

2.2 Seasonal adjustment examples

2.3 Manipulate workspaces

2.4 Installation and future development

2.5 Future developments

2.6 Statistics Canada dashboard

RegARIMA examples (1/4)

```
library(RJDemetra);options(enable_print_style = FALSE)
myseries <- ipi_c_eu[,"FR"]
regarima_model <- regarima_def_x13(myseries, spec = "RG4c")
str(regarima_model, max.level = 1)
```

```
## List of 9
##  $ specification          :List of 7
##  $ arma                   : Named num [1:6] 2 1 1 0 1 1
##   ..- attr(*, "names")= chr [1:6] "p" "d" "q" "bp" ...
##  $ arima.coefficients     : num [1:4, 1:3] 0.336 0.206 -0.245 -0.511 0.171 ..
##   ..- attr(*, "dimnames")=List of 2
##  $ regression.coefficients: num [1:4, 1:3] -1.133 -8 -7.551 -5.069 0.337 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ loglik                 : num [1:7, 1] -631 9 323 1280 1281 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ model                  :List of 2
##  $ residuals              : Time-Series [1:323] from 1991 to 2018: -2.601 0.5
##  $ residuals.stat         :List of 2
##  $ forecast               : Time-Series [1:24, 1:2] from 2018 to 2020: 102 10
##   ..- attr(*, "dimnames")=List of 2
##  - attr(*, "class")= chr [1:2] "regarima" "X13"
```

RegARIMA examples (2/4)

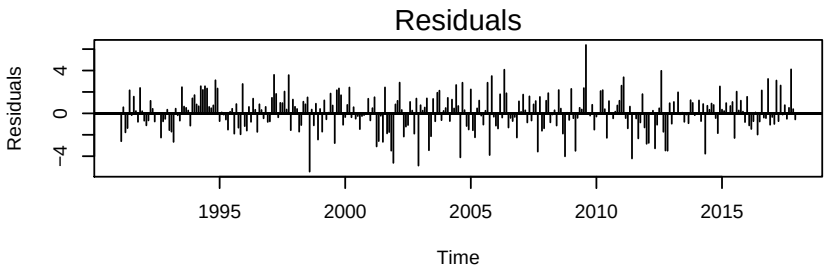
Console | Terminal | R Markdown

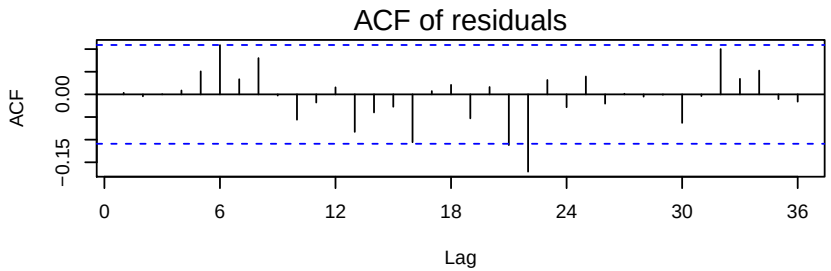
~/NTTS/

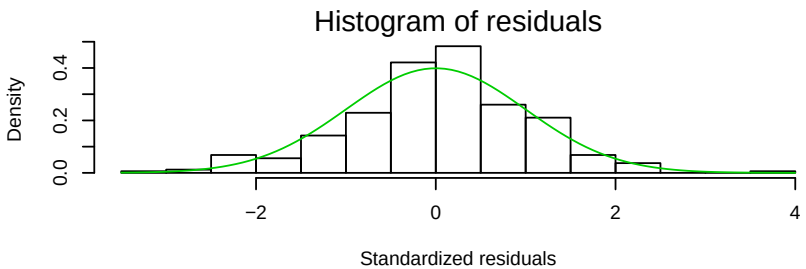
```
> regarima_model
y = regression model + arima (2, 1, 1, 0, 1, 1)
Log-transformation: no
Coefficients:
          Estimate Std. Error
Phi(1)      0.3358      0.171
Phi(2)      0.2060      0.096
Theta(1)   -0.2450      0.173
BTheta(1)  -0.5112      0.050

             Estimate Std. Error
Easter [1]     -1.133      0.337
LS (11-2008)   -8.000      1.283
LS (1-2009)    -7.551      1.283
LS (5-2008)    -5.069      1.234


Residual standard error: 1.696 on 323 degrees of freedom
Log likelihood =  -631, aic =  1280 aicc =  1281, bic(corrected for length) =   1.2

>
```

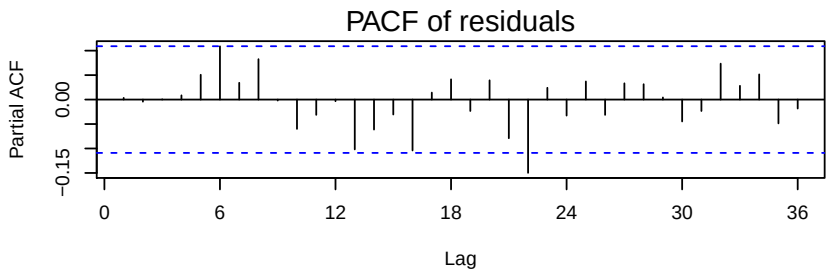
RegARIMA examples (3/4)

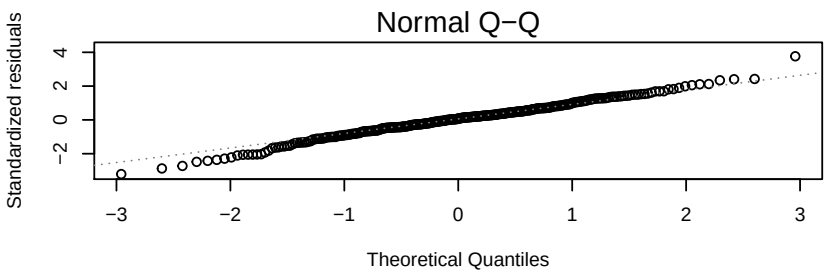
```
layout(matrix(1:6, 3, 2));plot(regarima_model, ask = FALSE)
```

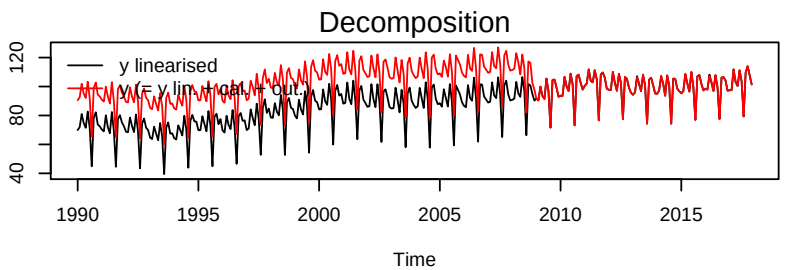






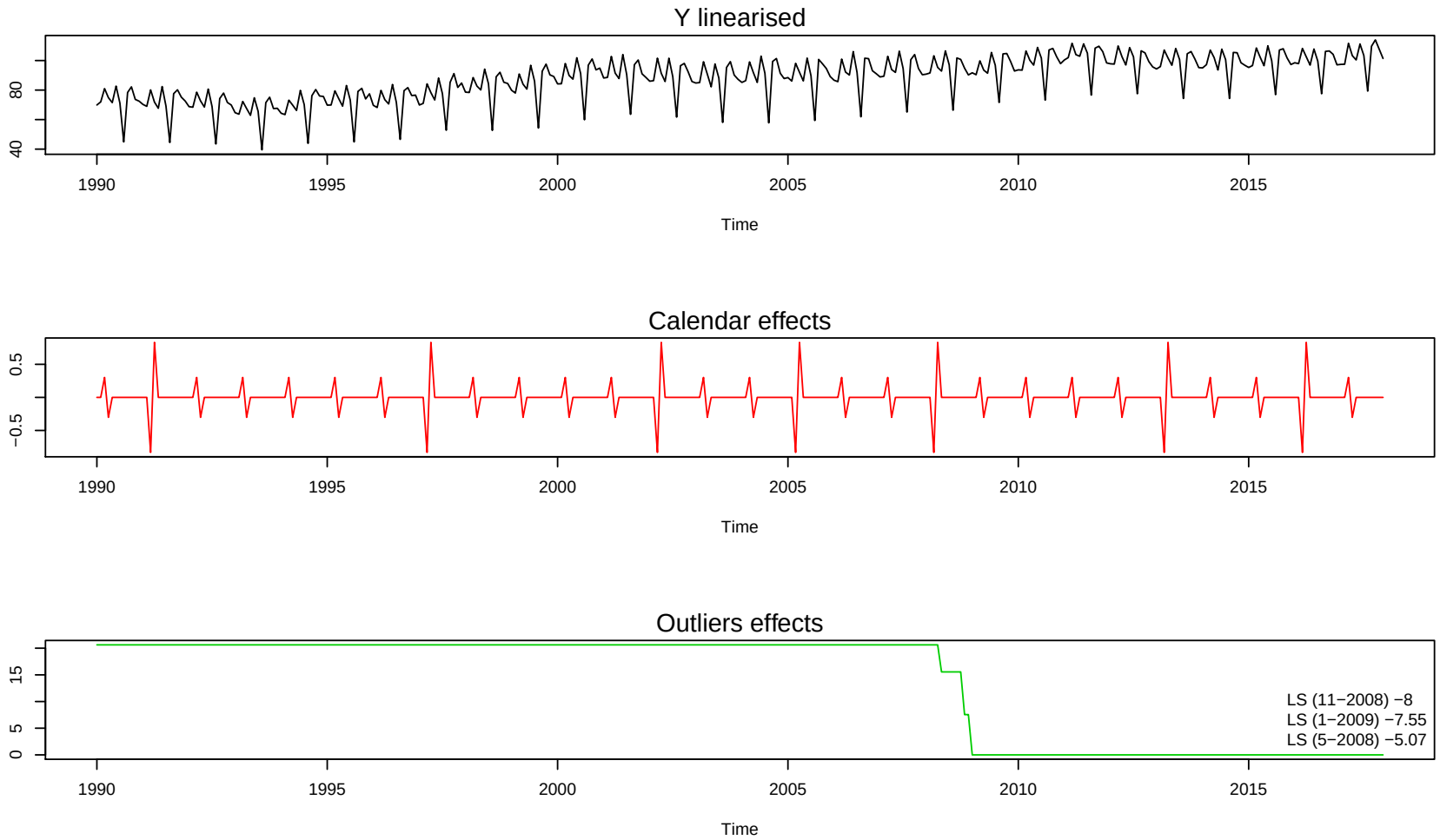






RegARIMA examples (3/4): extra plot

```
plot(regarima_model, which = 7, 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)     0.061
## M(2)     0.000
## M(3)     0.960
## M(4)     0.621
## M(5)     0.725
## M(6)     0.244
## M(7)     0.075
## M(8)     0.216
## M(9)     0.055
## M(10)    0.175
## M(11)    0.160
## Q        0.302
## Q-M2     0.339
##
## Final filters:
## Seasonal filter:  3x5
## Trend filter:  13 terms Henderson moving average
```

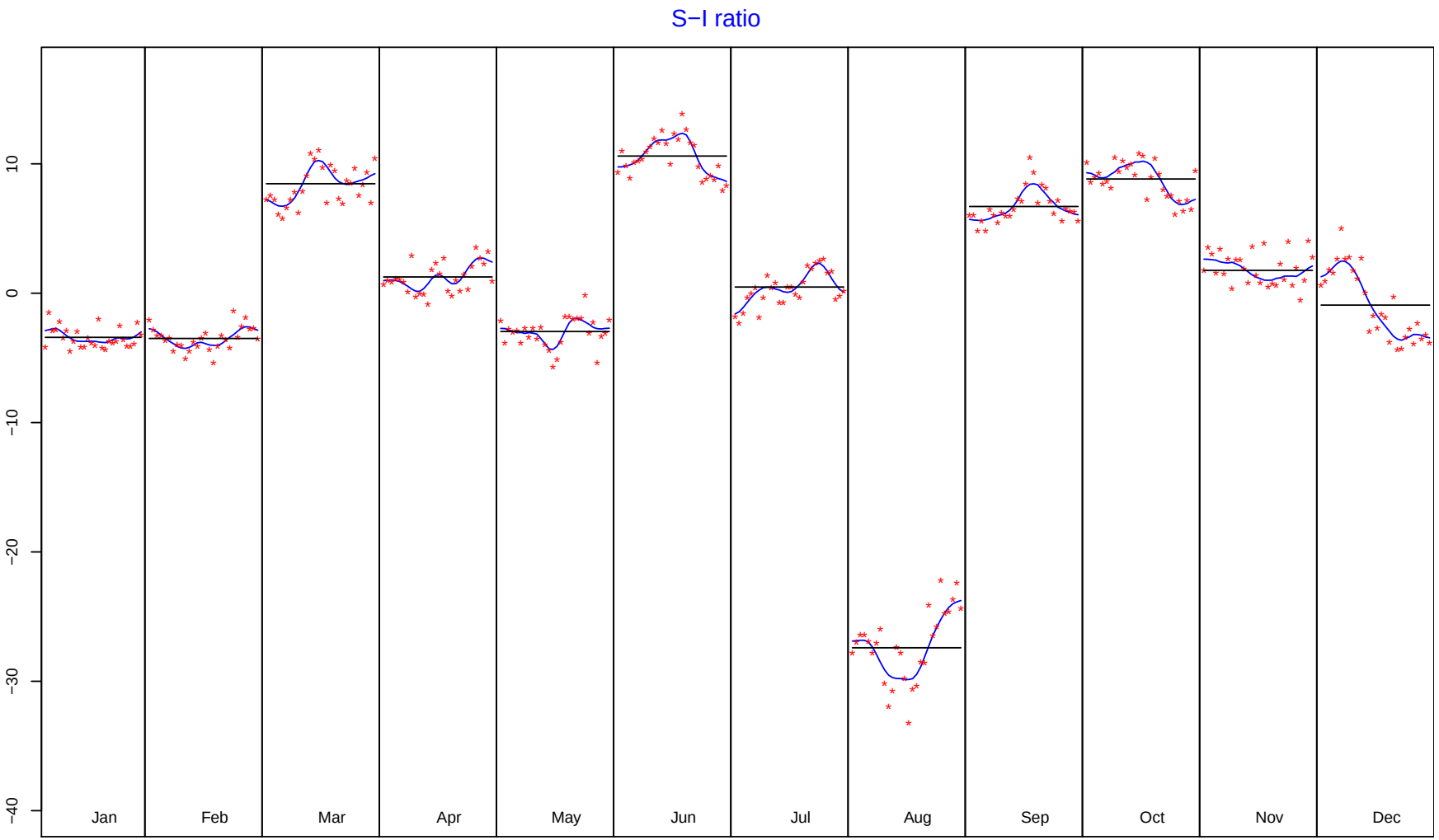
Seasonal adjustment examples (3/7)

```
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 (4/7)

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



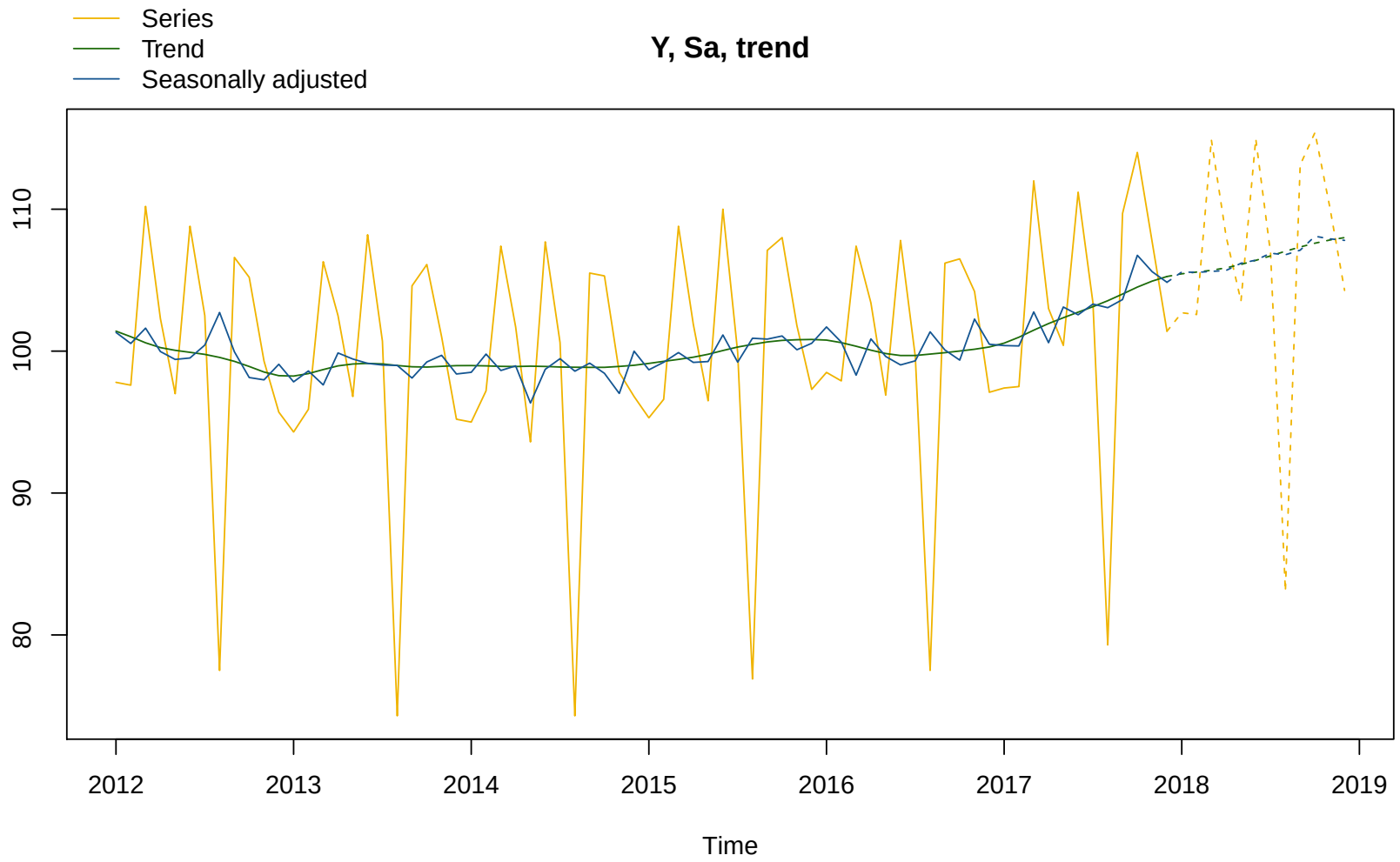
Seasonal adjustment examples (5/7)

```
x13_mod$final
```

```
## Last observed values
##                y       sa        t             s          i
## Jan 2017  97.4 100.3902 100.5631  -2.99019403 -0.1729099
## Feb 2017  97.5 100.3646 100.9790  -2.86460513 -0.6143814
## Mar 2017 112.0 102.7607 101.4754   9.23928088  1.2852852
## Apr 2017 103.0 100.5925 101.9417   2.40747193 -1.3491871
## May 2017 100.4 103.1054 102.3511  -2.70539979  0.7542873
## Jun 2017 111.2 102.5591 102.7440   8.64087923 -0.1848857
## Jul 2017 103.4 103.3144 103.1308   0.08560672  0.1835989
## Aug 2017  79.3 103.0602 103.5499 -23.76018864 -0.4897563
## Sep 2017 109.7 103.6331 104.0215   6.06692870 -0.3884680
## Oct 2017 114.0 106.7526 104.5113   7.24741395  2.2412439
## Nov 2017 107.7 105.5963 104.9317   2.10371268  0.6645400
## Dec 2017 101.4 104.8511 105.2567  -3.45110589 -0.4056223
##
## Forecasts:
##                  y_f     sa_f      t_f           s_f          i_f
## Jan 2018 102.70258 105.5740 105.4465  -2.87143758  0.12749286
## Feb 2018 102.57727 105.5466 105.5700  -2.96928512 -0.02343246
## Mar 2018 114.89223 105.6137 105.6941   9.27854825 -0.08039957
## Apr 2018 108.05979 105.6858 105.8691   2.37398629 -0.18326092
```

Seasonal adjustment examples (6/7)

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



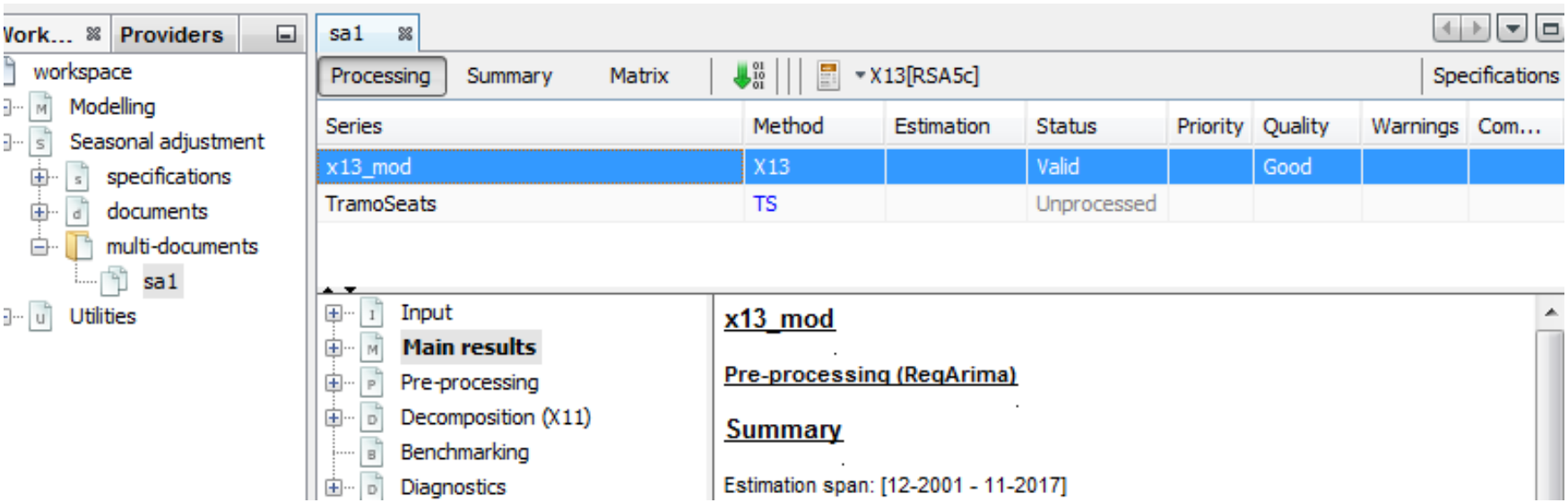
Seasonal adjustment examples (7/7)

```
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          1.656
##  Seasonal      39.710
##  Irregular      0.369
##  TD & Hol.      0.000
##  Others        61.757
##  Total        103.492
##
##  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                                     0.024
##
##  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")
```

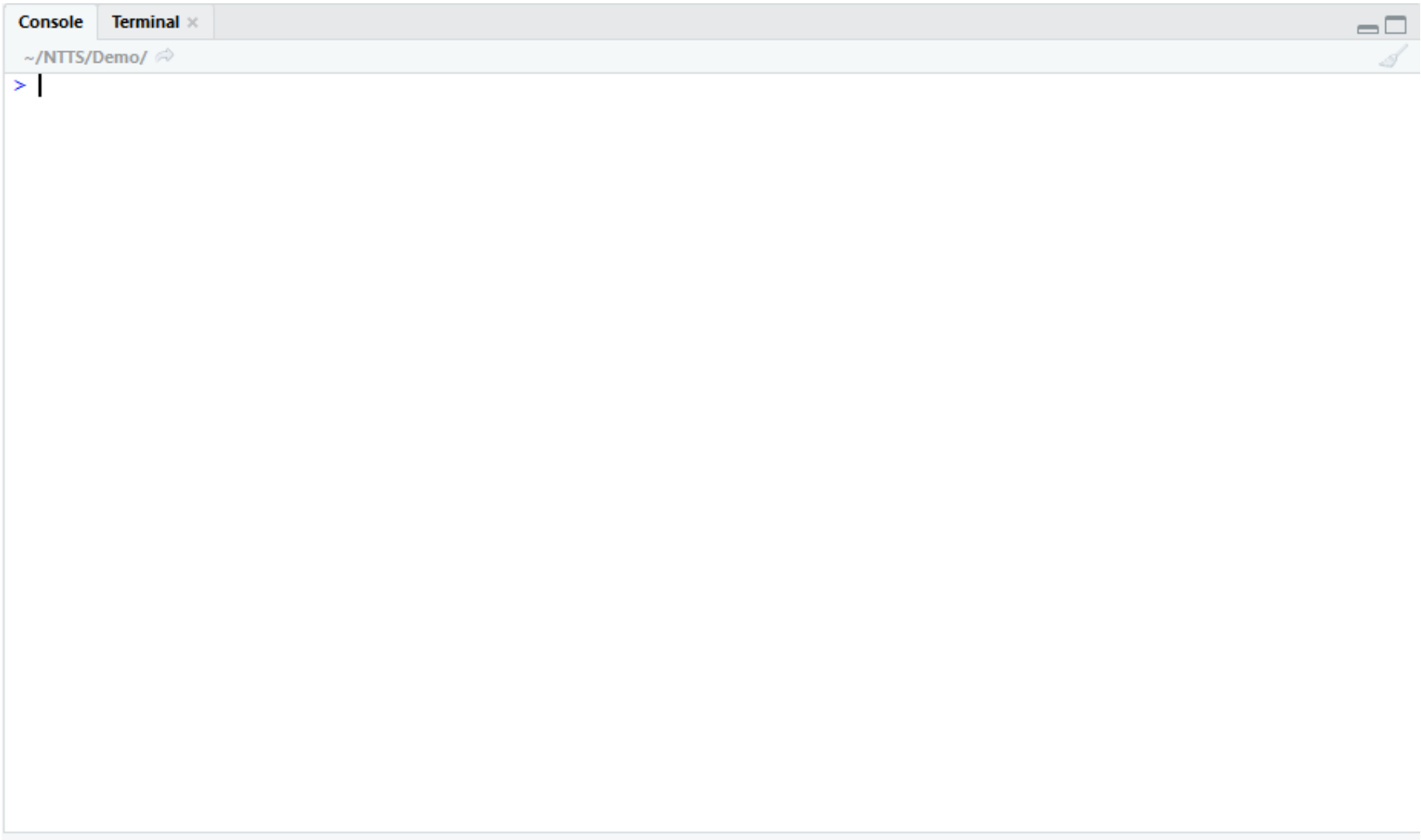


Import a workspace (1/3)

```
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)



Import a workspace (3/3)

```
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)
```

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)

Why and how use RJDemetra?

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

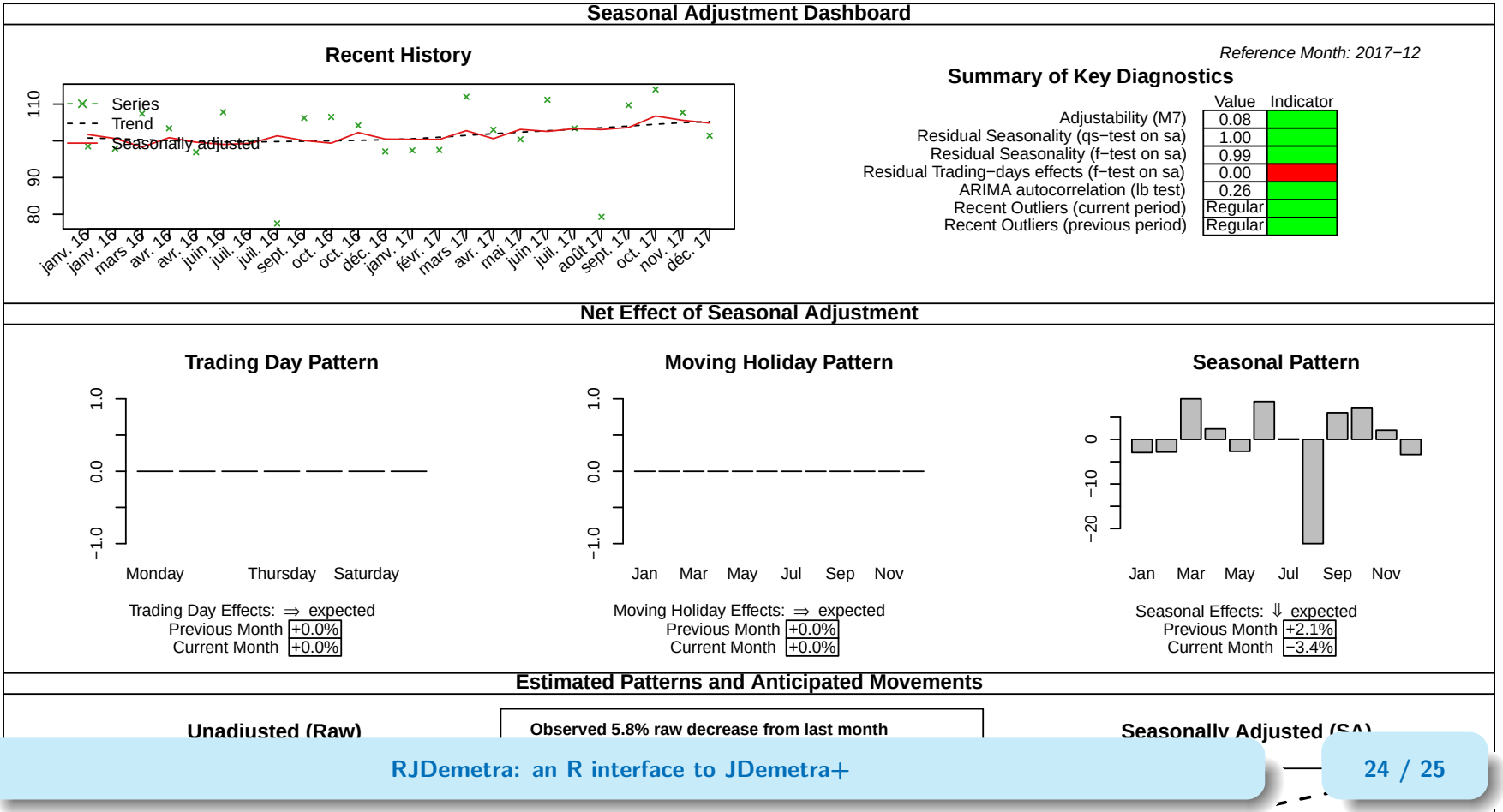
Available on github https://github.com/AQLT/rjdqa, still in development (only works for X13 models and no documentation yet)

Example of the dashboard:

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

Example of the dashboard (1/2)

```
##
## Attaching package: 'rjdqa'

## The following object is masked _by_ '.GlobalEnv':
##
##     myseries
```



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 $SA_t = SA_{t-1}$)