BCEAO - 20 au 25 janvier 2019



# 1 - R et JDemetra+

Dominique Ladiray et Alain Quartier-la-Tente

# Sommaire

# Le JWSACruncher

Objectifs du cruncher : mettre à jour un workspace de JDemetra+ et exporter les résultats à partir de la console (en *batch*), sans devoir ouvrir JDemetra+ : très utile pour la production. Quelques liens :

- pour télécharger le cruncher https://github.com/jdemetra/jwsacruncher/releases.
- l'aide associée au cruncher https://github.com/jdemetra/jwsacruncher/wiki.
- configuration du cruncher une version portable de Java : https://github.com/AQLT/JDCruncheR/wiki/Installation-et-configuration-de-JDemetra--et-du-cruncher.

# Le cruncher

Pour lancer le cruncher de JDemetra+ il faut :

- le cruncher ;
- un fichier contenant les paramètres sur la méthode de rafraîchissement à utilisée pour mettre à jour le workspace et sur les paramètres d'export ;
- un workspace valide de JDemetra+.

# Installation du package

Le package `rjwsacruncher` est une interface autour du JWSACruncher.

Il est disponible sur le CRAN a une page GitHub associée : https://github.com/AQLT/rjwsacruncher.

```
install.packages("rjwsacruncher")
```

# Utilisation de JDCruncheR (1/3)

Une vignette décrit plus précisément la procédure pour utiliser le cruncher à partir du package :

```
browseVignettes("rjwsacruncher")
```

Pour charger le package :

```
library(JDCruncheR)
```

# Utilisation de JDCruncheR (2/3)

Trois options vont être utiles : `default_matrix_item` (diagnostics à exporter), `default_tsmatrix_series` (séries temporelles à exporter) et `cruncher_bin_directory` (chemin vers le cruncher).

Pour afficher les valeurs :

```
getOption("default_matrix_item")
getOption("default_tsmatrix_series")
getOption("cruncher_bin_directory")
```

Utiliser la fonction `options()` pour les modifier. Par exemple :

```
options(default_matrix_item = c("likelihood.aic",
                                "likelihood.aicc",
                                "likelihood.bic",
                                "likelihood.bicc"))
options(default_tsmatrix_series = c("sa", "sa_f"))
options(cruncher_bin_directory =
            "Y:/Logiciels/jwsacruncher-2.2.0/jdemetra-cli-2.2.0/b
```

# Utilisation de JDCruncheR (3/3)

Une fois les trois options précédentes validées le plus simple est d'utiliser la fonction `cruncher_and_param()` :

```
cruncher_and_param() # lancement avec paramètres par défaut

cruncher_and_param(workspace = "D:/Campagne_CVS/ipi.xml",
                   policy = "lastoutliers")
```

Pour voir l'aide associée à une fonction, utiliser `help()` ou `?` :

```
?cruncher_and_param
help(cruncher_and_param)
```

$\longrightarrow$ Dans le TP le cruncher sera lancé en créant un fichier de paramètres

# Sommaire

# RJDemetra


RJDemetra est un package qui permet de lancer les routines de JDemetra+ depuis R

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

Page web : https://jdemetra.github.io/rjdemetra/

Pour l'installer :

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

→ Peut être utilisé pour développer de nouveaux outils pour aider la production

→ Il faut Java 8 ou plus pour l'utiliser. En cas de problème d'installation : https://github.com/jdemetra/rjdemetra/wiki/Installation-manual

# Current status

- RegARIMA, TRAMO-SEATS et X-13-ARIMA :
  - spécifications prédéfinies et personnalisées
  - classes S3 avec des méthodes plot, summary, print
- Manipulation de workspaces JD+ :
  - Import de workspaces to avec le modèle CVS
  - Export des modèles R créé par RJDemetra
- Contient une base de données (`ìpi_c_eu`) : les IPI dans l'industrie manufacturière dans l'UE

# RegARIMA : exemples (1/4)

```
library(RJDemetra)
ipi_fr <- ipi_c_eu[,"FR"]
regarima_model <- regarima_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
```

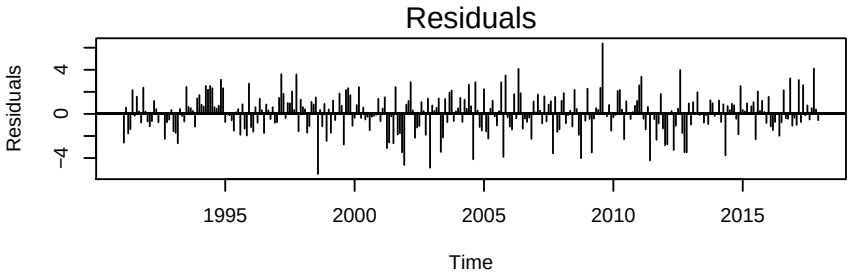
# RegARIMA : exemples (2/4)

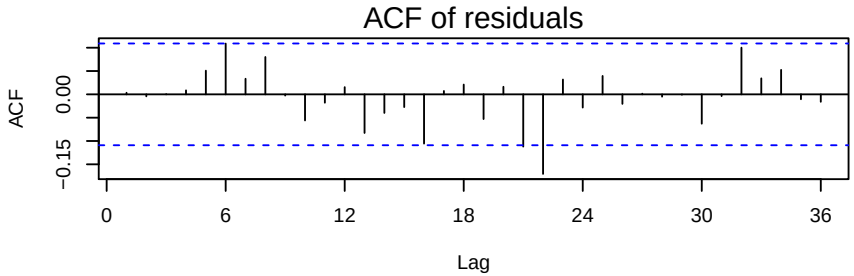
```
summary(regarima_model)
```

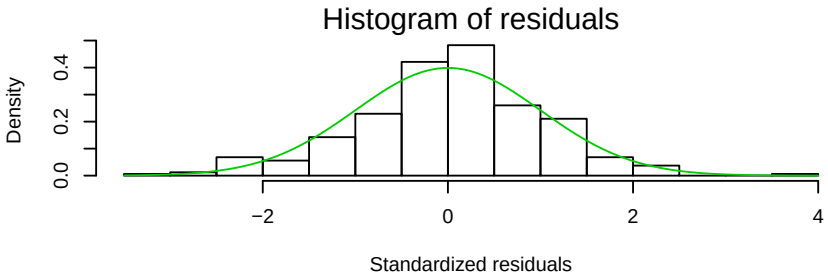
```
## y = regression model + arima (2, 1, 1, 0, 1, 1)
##
## Model: RegARIMA - X13
## Estimation span: from 1-1990 to 12-2017
## Log-transformation: no
## Regression model: no mean, no trading days effect, no leap year effect, Easte
##
## Coefficients:
## ARIMA:
##           Estimate Std. Error  T-stat 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 ***
```

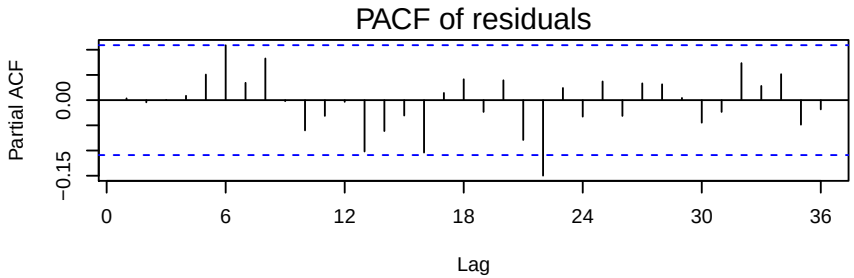
# RegARIMA : exemples (3/4)

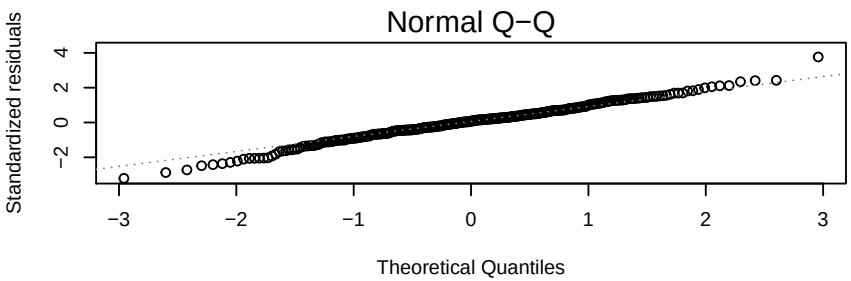
```
layout(matrix(1:6, 3, 2));plot(regarima_model, ask = FALSE)
```

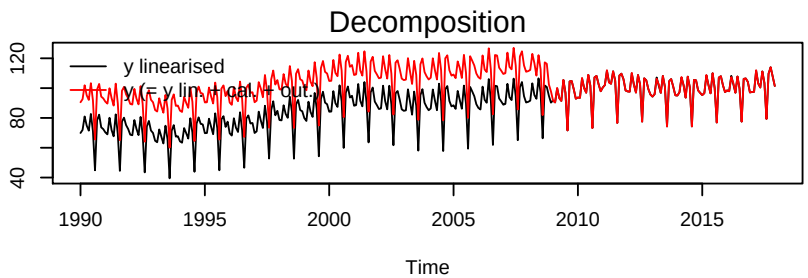


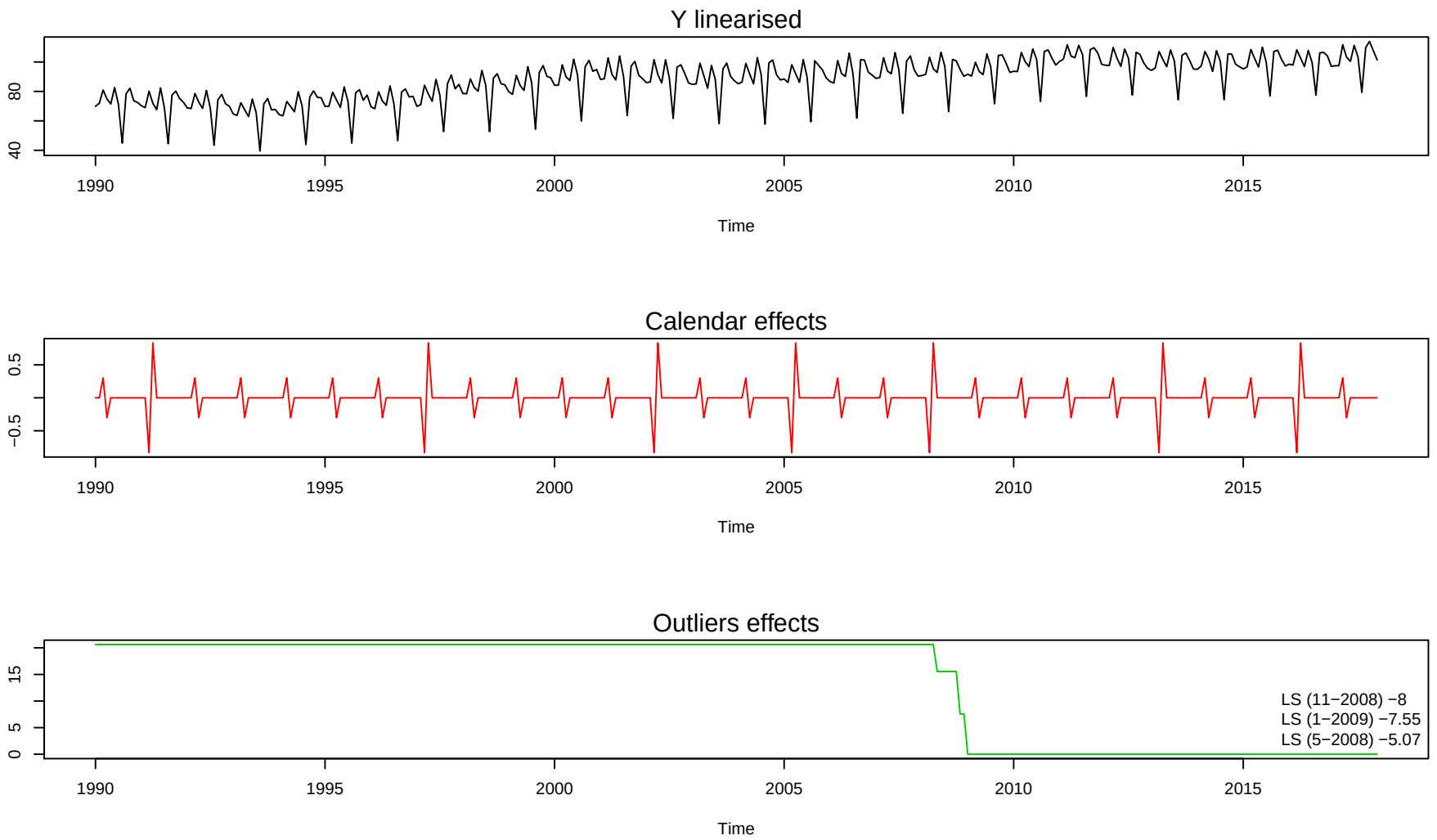



# RegARIMA : exemples (4/4)

```
plot(regarima_model, which = 7)
```



# CVS-CJO : exemples (1/8)

Un object `SA` est une `list()` de 5 éléments :

```
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
```

# CVS-CJO : exemples (2/8)

Possibilité de définir ses propres spécifications comme sous JD+ ou d'utiliser les spécifications prédéfinies :

```
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)
ts_mod <- tramoseats(ipi_fr, spec = "RSAfull")
```

# CVS-CJO : exemples (3/8) : 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-Henderson
```

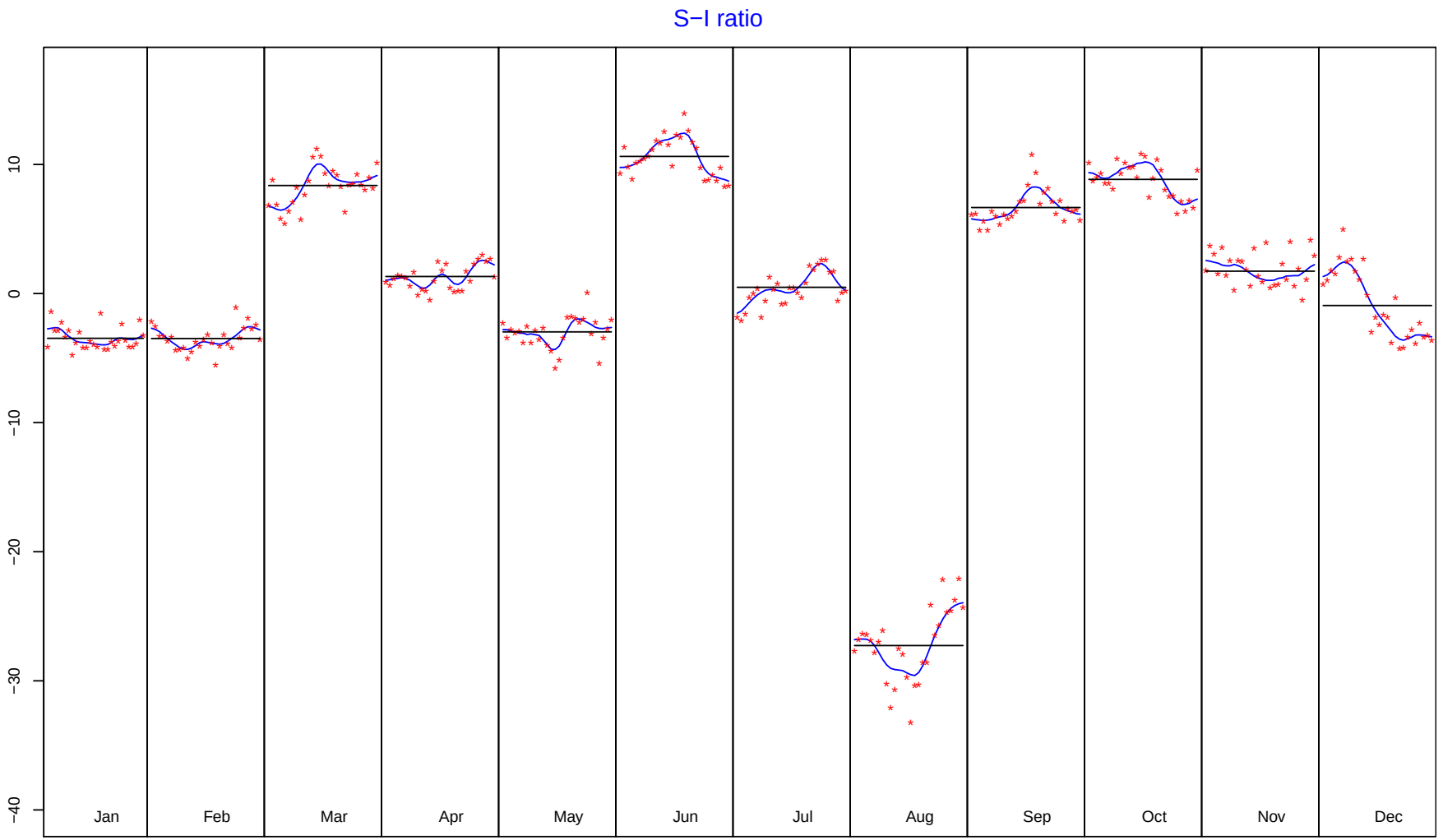
# CVS-CJO : exemples (4/8) : decomposition

```
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
```

# CVS-CJO : exemples (5/8)

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



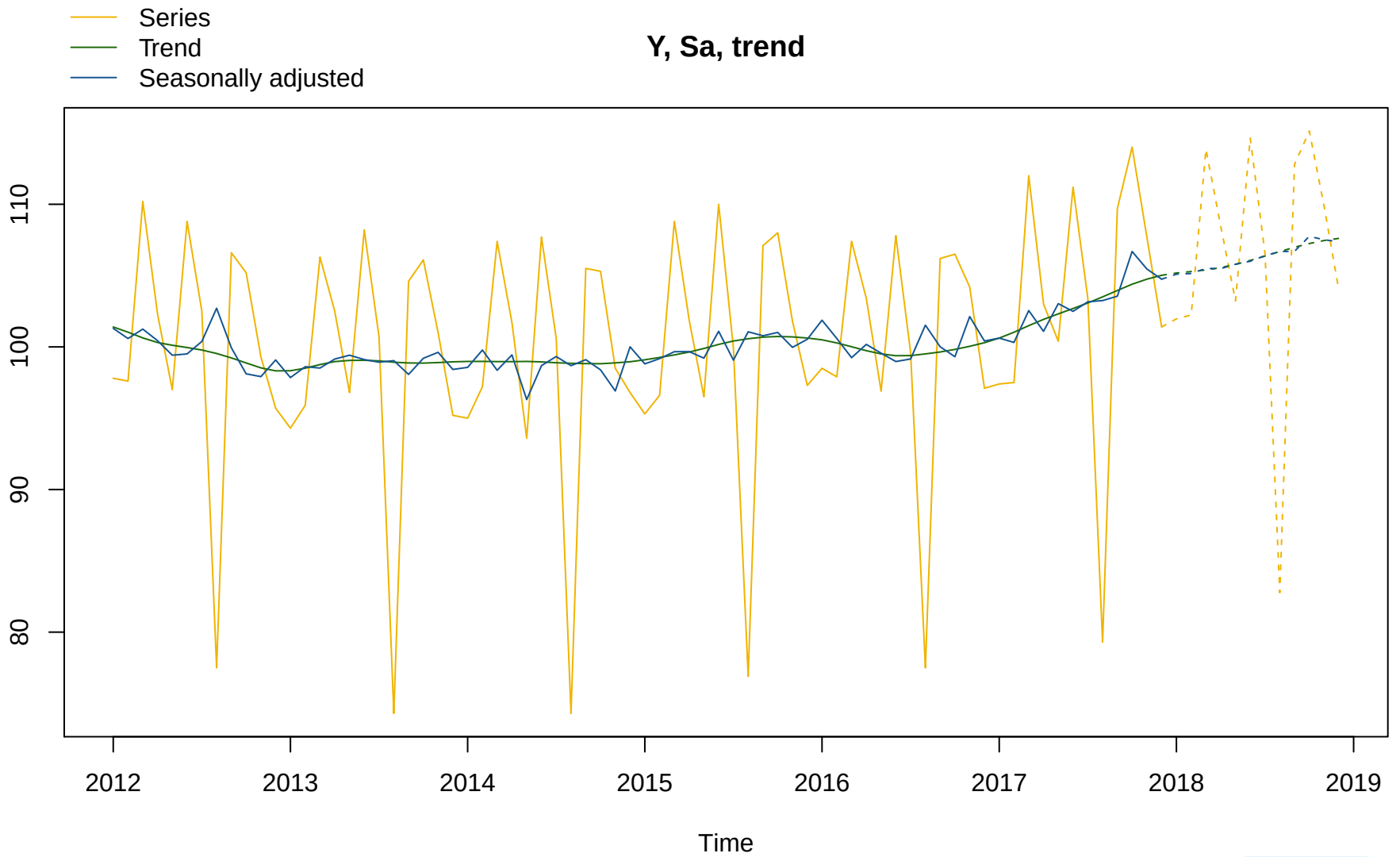
# CVS-CJO : exemples (6/8)

```
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
```

# CVS-CJO : exemples (7/8)

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



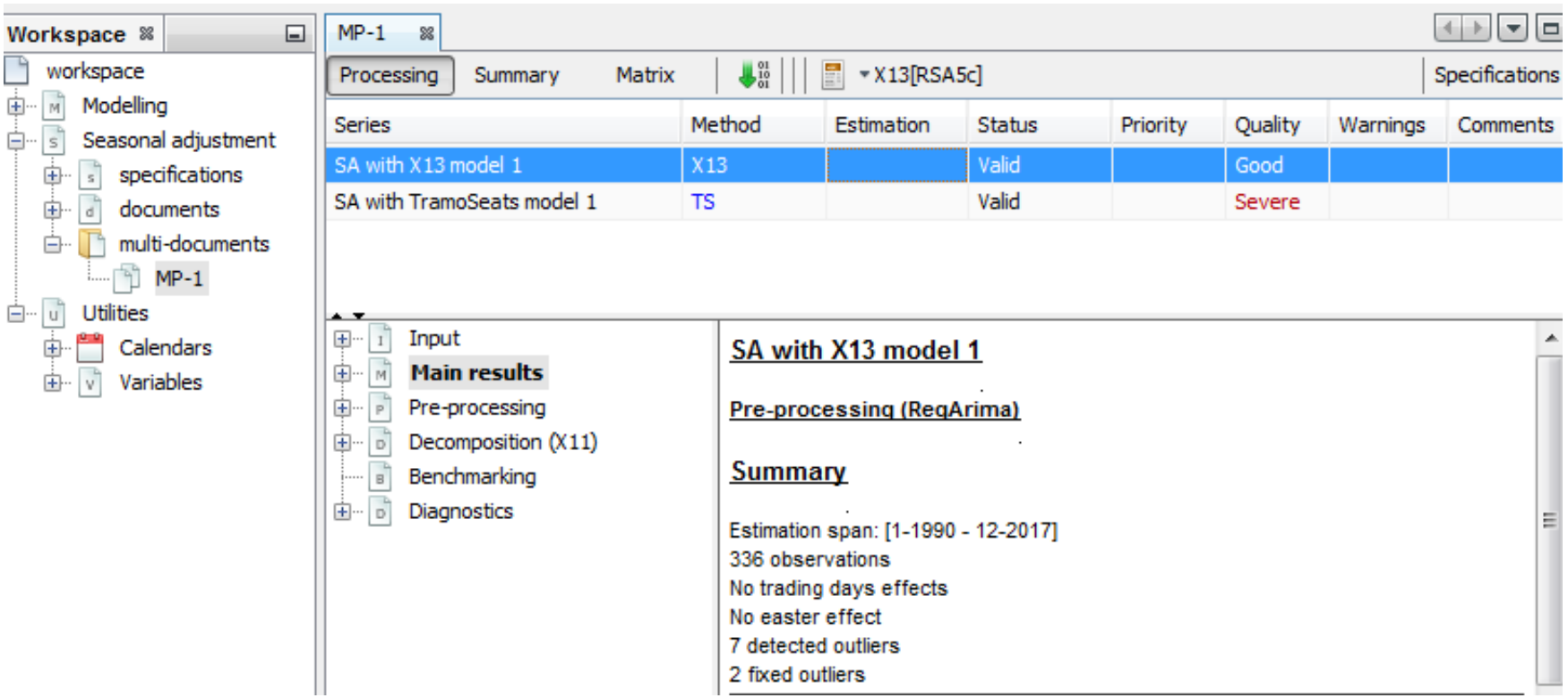
# CVS-CJO : exemples (8/8)

```
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
##
```

# Exporter un 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")
```

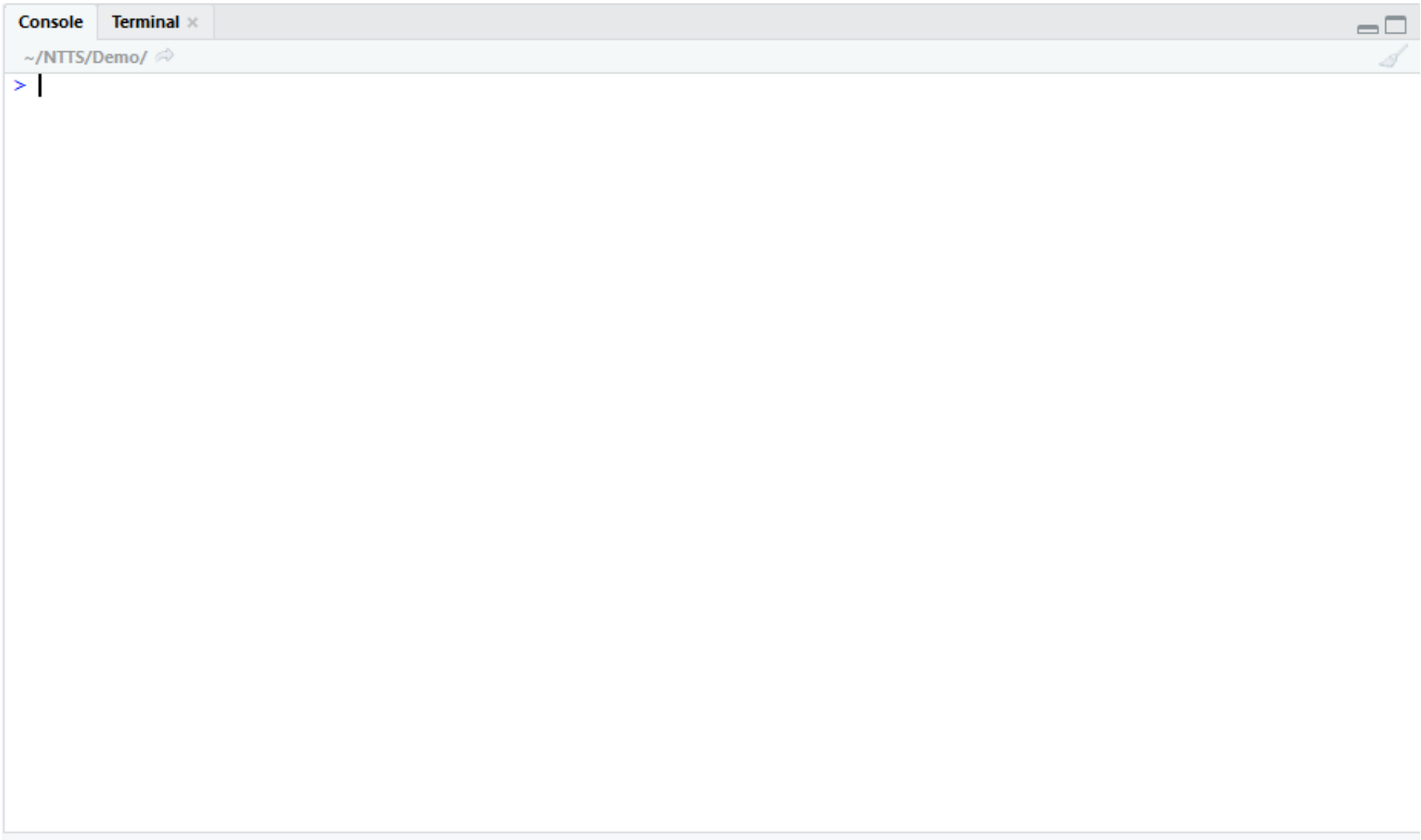


# Importer un workspace (1/3)

```
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   Dec
## 1990  90.5  92.6 101.9  95.2  92.1 103.3  91.8  65.5  99.0 102.8  94.3  93.1
## 1991  90.9  89.6  99.9  93.3  88.3 103.0  89.7  65.1  98.2 100.8  95.8  93.2
## 1992  89.4  89.0  99.5  93.0  89.1 101.3  89.4  64.1  94.9  98.6  92.2  90.5
## 1993  85.3  84.3  93.2  87.8  83.5  95.4  86.2  60.1  92.1  95.8  88.1  88.3
## 1994  84.9  84.0  94.1  90.1  86.8 100.4  90.8  64.5  96.8 101.0  96.6  96.3
## 1995  90.4  90.5 100.4  94.5  89.7 103.7  93.8  65.5  99.7 101.8  94.6  98.1
## 1996  90.3  88.8 100.7  93.8  91.2 104.4  92.3  67.2 100.2 102.3  96.9  97.2
## 1997  90.5  91.6 104.0  99.7  93.9 108.8  98.2  73.4 105.8 111.8 102.4 105.4
## 1998  99.2  99.0 109.4 103.0 100.7 114.8 104.9  73.3 109.6 112.7 105.9 105.1
## 1999 100.5  98.6 111.8 104.3 101.3 117.4 106.6  74.9 113.4 118.2 110.9 109.8
## 2000 104.8 104.9 118.9 110.2 108.0 122.5 111.8  80.5 117.5 121.7 114.3 115.5
## 2001 108.8 109.2 123.7 111.8 108.4 124.7 111.1  84.2 117.8 121.0 111.6 109.2
## 2002 106.6 107.0 121.4 112.8 106.4 122.2 109.7  82.3 117.1 118.7 113.0 106.4
## 2003 105.4 105.7 120.1 111.1 102.8 118.3 108.8  78.7 115.9 119.9 110.8 107.9
## 2004 105.8 107.0 120.0 112.1 105.8 123.6 112.0  78.4 120.0 122.0 112.0 108.4
## 2005 109.1 106.7 117.9 113.5 106.8 122.3 110.3  80.0 121.4 118.4 115.2 109.8
## 2006 107.3 106.3 121.9 112.5 110.8 126.7 112.5  82.5 122.2 121.9 113.7 111.7
```

# Importer un workspace (2/3)



# Importer un 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)
```

# En manipulant les objets objects (1/2)

Les fonctions de base peuvent être chronophages (calcul de tous les outpus)...
Notamment lorsqu'on ne s'intéresse qu'à un seul paramètre (série désaisonnalisée, tendance, etc.)

→ Solution : manipuler les objets Java : `jx13`, `jtramoseats`, `jregarima`, `jregarima_x13`, `jregarima_tramoseats` and `get_jmodel`

# En manipulant les objets ☕ objects (1/2)

Les fonctions de base peuvent être chronophages (calcul de tous les outpus)...
Notamment lorsqu'on ne s'intéresse qu'à un seul paramètre (série désaisonnalisée, tendance, etc.)

→ Solution : manipuler les objets Java : `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)
```

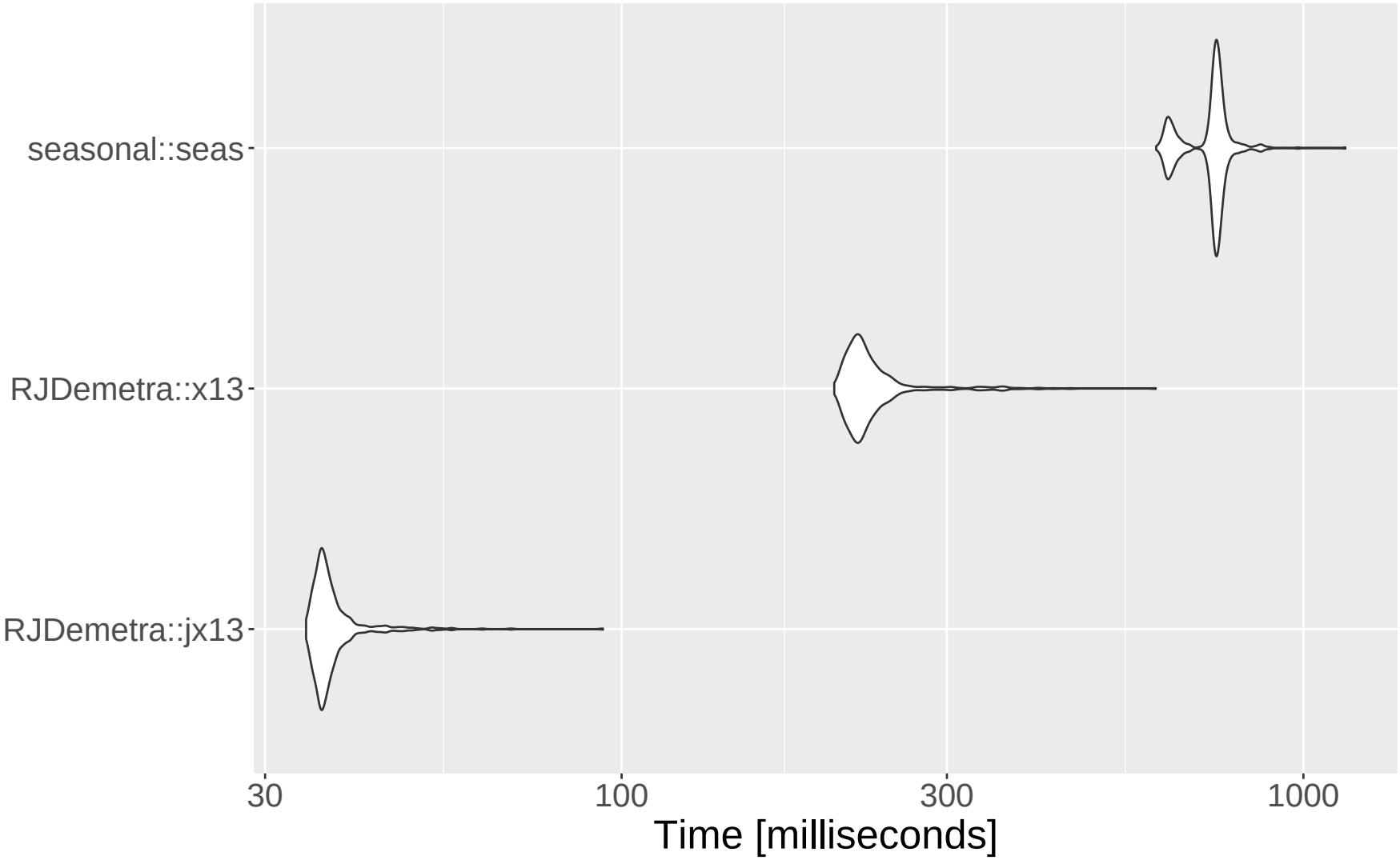
```
## [1] "diagnostics.msr-global" "diagnostics.msr(*)"
```

```
# 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)
```

# Performance



# Exemples d'utilisation de RJDemetra

- rjdqa (sur le CRAN) : package pour aider à évaluer la qualité de la désaisonnalisation (tableau de bord et bientôt tests de saisonnalité)

https://github.com/AQLT/rjdqa

- ggdemetra (sur le CRAN) : intégrer la désaisonnalisation à ggplot2

https://github.com/AQLT/ggdemetra

- rjdworkspace (en développement) : fonctions supplémentaires pour manipuler les workspaces

https://github.com/AQLT/rjdworkspace

- rjdmarkdown (en développement) : pour exporter directement les modèles en PDF/HTML

https://github.com/AQLT/rjdmarkdown

- Réalisations d'études : Ladiray D., Quartier-la-Tente A., "Du bon usage des modèles Reg-ARIMA en désaisonnalisation", JMS 2018