BCEAO - 20 au 25 janvier 2019

1 - R et JDemetra+

Dominique Ladiray et Alain Quartier-la-Tente
(dominique.ladiray@insee.fr et alain.quartier@yahoo.fr)

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 rjwsacruncher (1/3)

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

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

Pour charger le package :

```
library(rjwsacruncher)
```

Utilisation de rjwsacruncher (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 rjwsacruncher (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 (`ipi_c_eu`) : les IPI dans l'industrie manufacturière dans l'UE

RegARIMA : exemples (1/4)

```
library(RJDemetra)
ipch_benin <- ipch_benin[,"ensemble"]
regarima_model <- regarima_x13(ipch_benin, spec = "RG4c")
regarima_model
```

```
## y = regression model + arima (0, 1, 0, 0, 1, 1)
## Log-transformation: no
## Coefficients:
##           Estimate Std. Error
## BTheta(1)  -0.9145      0.036
##
##              Estimate Std. Error
## LS (1-2012)     4.379      0.824
## AO (4-2015)    -2.747      0.587
## AO (6-2009)    -2.322      0.582
##
##
## Residual standard error: 0.8265 on 237 degrees of freedom
```

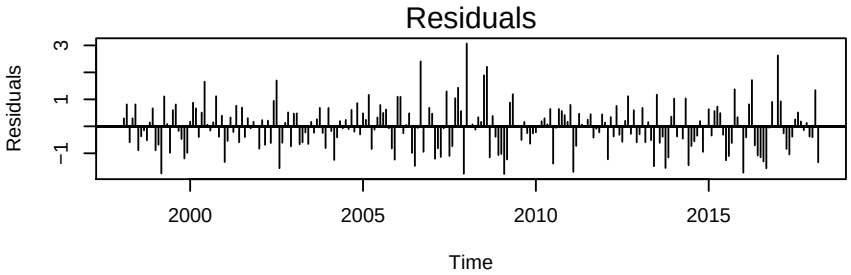
RegARIMA : exemples (2/4)

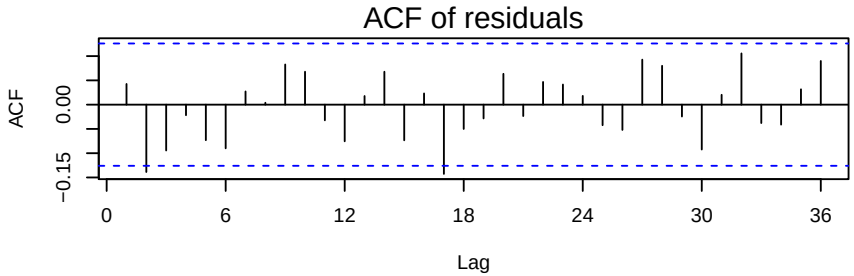
```
summary(regarima_model)
```

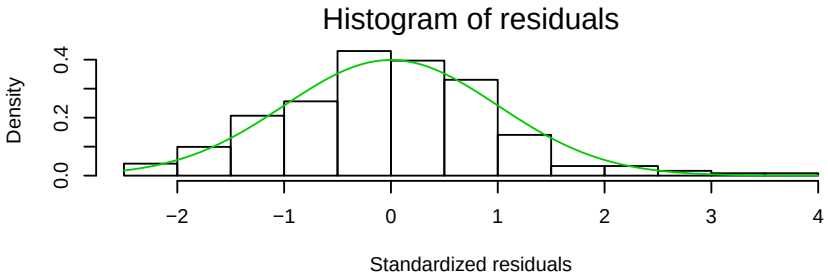
```
## y = regression model + arima (0, 1, 0, 0, 1, 1)
##
## Model: RegARIMA - X13
## Estimation span: from 1-1997 to 3-2018
## Log-transformation: no
## Regression model: no mean, no trading days effect, no leap year effect, no Ea
##
## Coefficients:
## ARIMA:
##           Estimate Std. Error T-stat Pr(>|t|)
## BTheta(1)  -0.9145     0.0355 -25.76   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Regression model:
##              Estimate Std. Error T-stat Pr(>|t|)
## LS (1-2012)   4.3790     0.8241  5.314 2.43e-07 ***
## AO (4-2015)  -2.7470     0.5869 -4.681 4.76e-06 ***
## AO (6-2009)  -2.3224     0.5818 -3.992 8.71e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

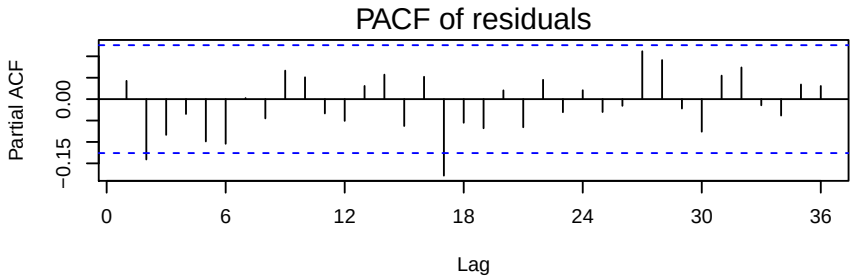
RegARIMA : exemples (3/4)

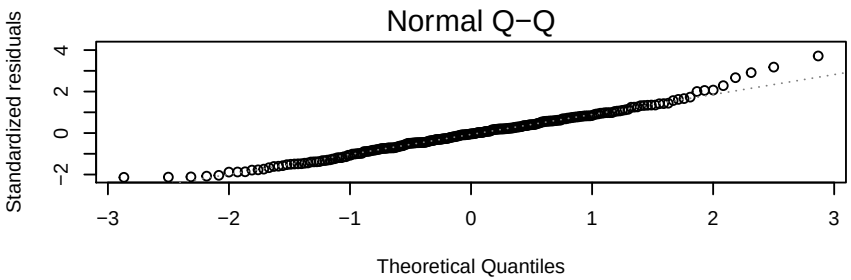
```
layout(matrix(1:6, 3, 2));plot(regarima_model, ask = FALSE)
```

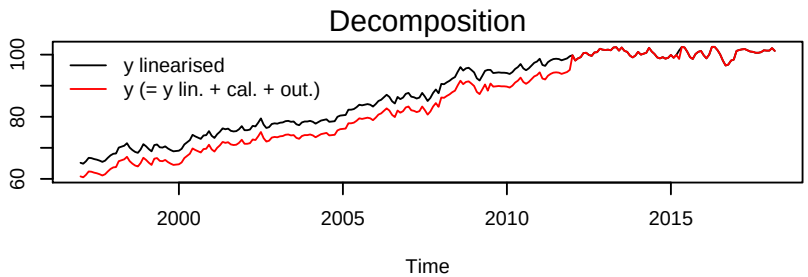






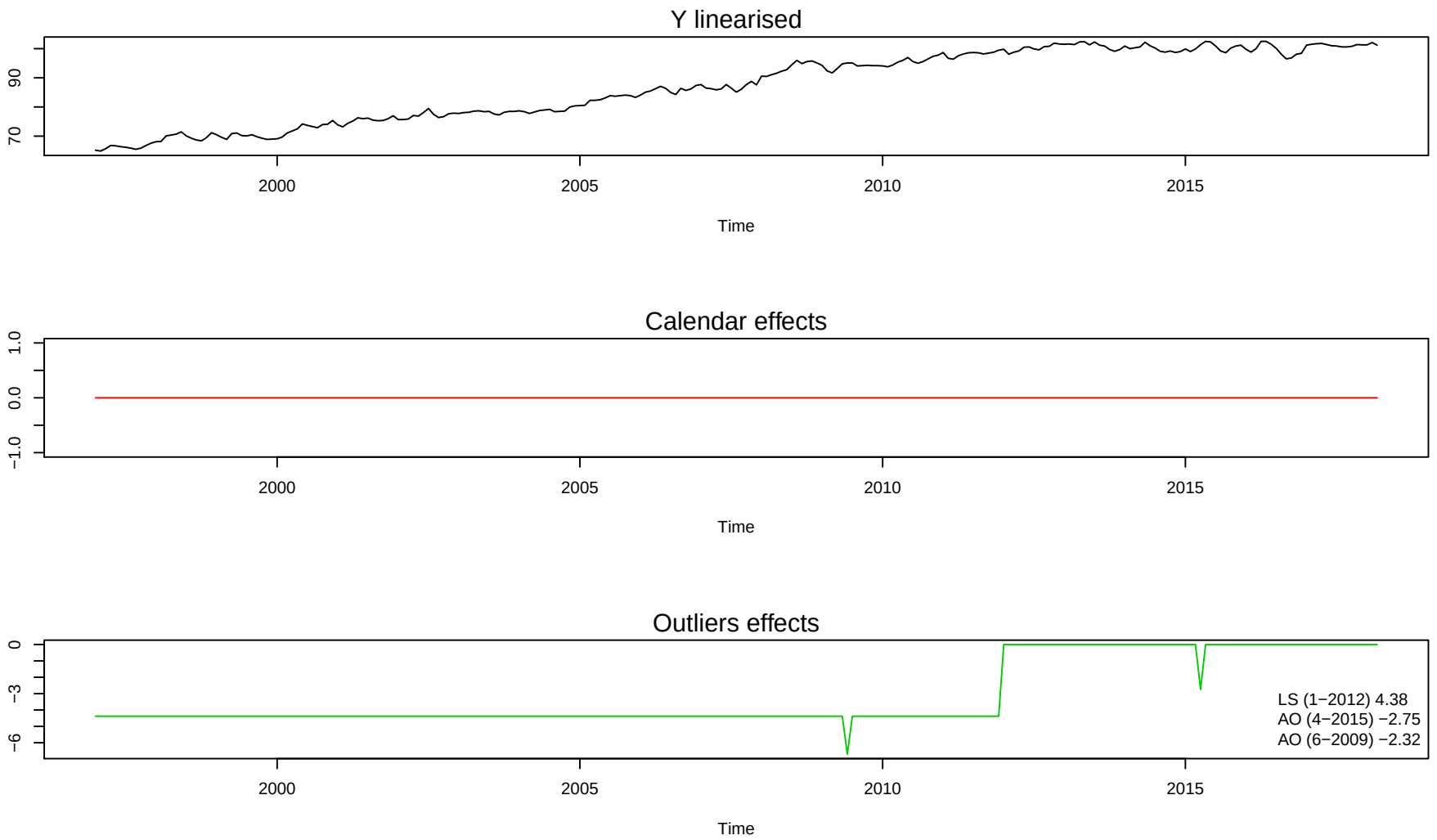






RegARIMA : exemples (4/4)

```
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(ipch_benin, x13_usr_spec)
ts_mod <- tramoseats(ipch_benin, spec = "RSAfull")
```

CVS-CJO : exemples (3/8) : decomposition

```
x13_mod$decomposition
```

```
##  Monitoring and Quality Assessment Statistics:
##        M stats
## M(1)     1.817
## M(2)     0.142
## M(3)     0.219
## M(4)     1.600
## M(5)     0.458
## M(6)     0.199
## M(7)     0.786
## M(8)     1.564
## M(9)     0.354
## M(10)    2.682
## M(11)    2.620
## Q        0.905
## Q-M2     1.000
##
## Final filters:
## Seasonal filter:  3x5
## Trend filter:  13-Henderson
```

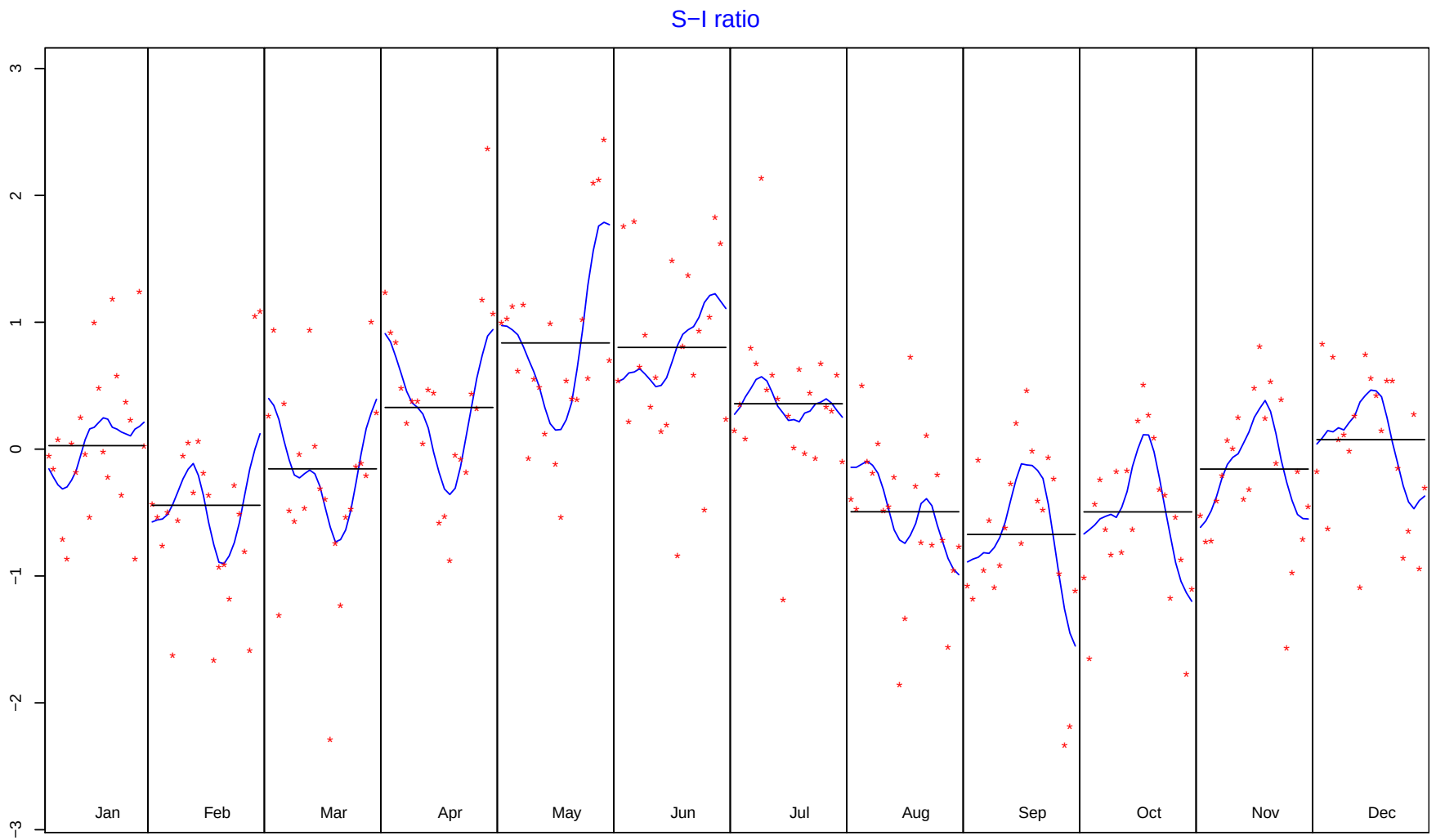
CVS-CJO : exemples (4/8) : decomposition

```
ts_mod$decomposition
```

```
## Model
## D :  1 - B - B^12 + B^13
## MA :  1 - 0.923007 B^12
##
##
## SA
## D :  1 - 2.000000 B + B^2
## MA :  1 - 0.993616 B + 0.000268 B^2
## Innovation variance:  0.9301857
##
## Trend
## D :  1 - 2.000000 B + B^2
## MA :  1 + 0.006654 B - 0.993346 B^2
## Innovation variance:  0.2324235
##
## Seasonal
## 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 + 1.840644 B + 2.192789 B^2 + 2.271402 B^3 + 2.121758 B^4 + 1.844037
## Innovation variance:  0.002063135
##
## Irregular
```

CVS-CJO : exemples (5/8)

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



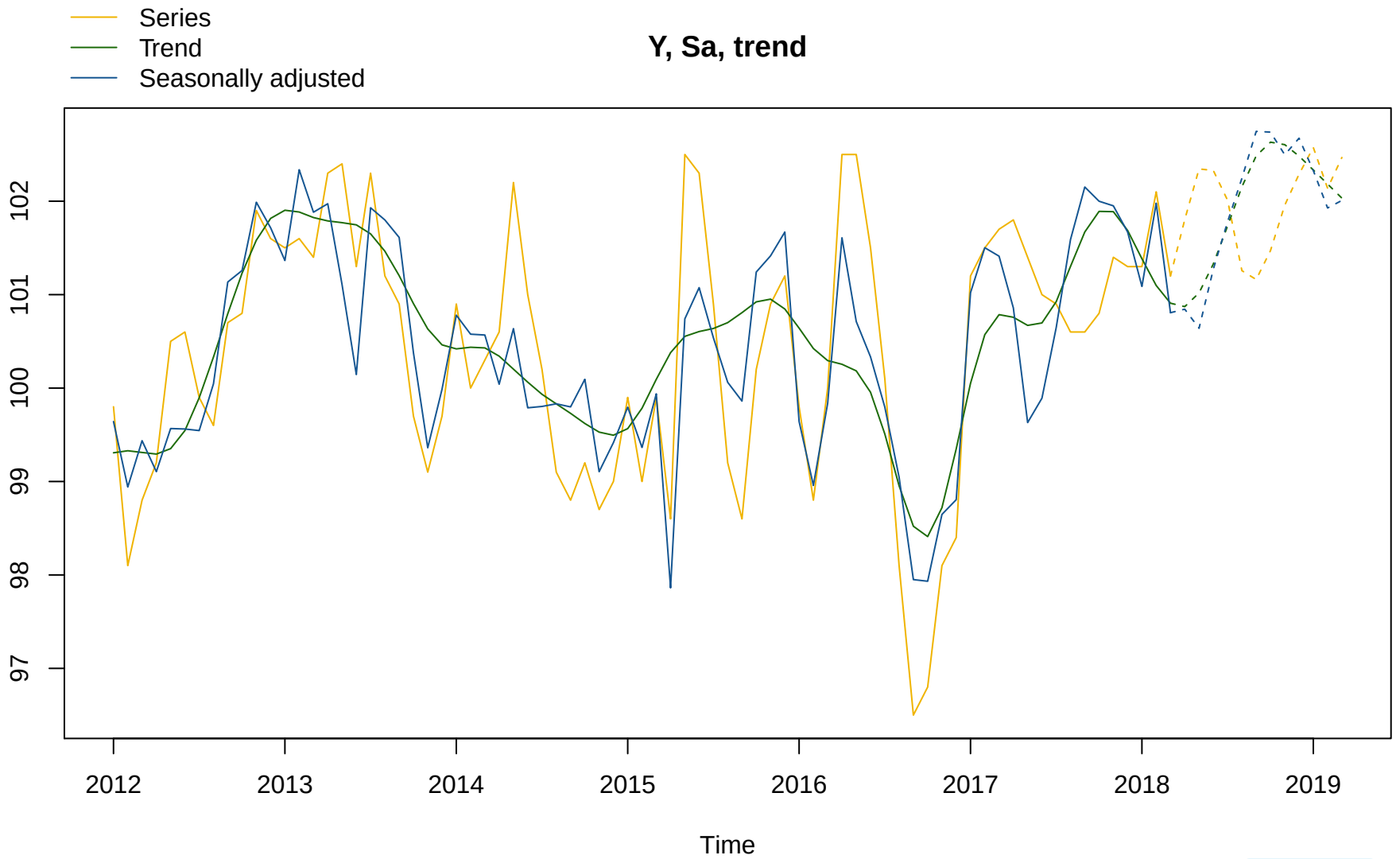
CVS-CJO : exemples (6/8)

```
x13_mod$final
```

```
## Last observed values
##                y        sa         t          s           i
## Apr 2017 101.8 100.85746 100.7578  0.9425435  0.09963373
## May 2017 101.4  99.63078 100.6708  1.7692187 -1.04001487
## Jun 2017 101.0  99.89153 100.6970  1.1084675 -0.80548123
## Jul 2017 100.9 100.64861 100.9250  0.2513894 -0.27642041
## Aug 2017 100.6 101.58999 101.3019 -0.9899894  0.28805653
## Sep 2017 100.6 102.15194 101.6705 -1.5519386  0.48143928
## Oct 2017 100.8 101.99956 101.8910 -1.1995588  0.10858753
## Nov 2017 101.4 101.95044 101.8879 -0.5504364  0.06254215
## Dec 2017 101.3 101.67004 101.6865 -0.3700423 -0.01650179
## Jan 2018 101.3 101.08775 101.3844  0.2122531 -0.29663316
## Feb 2018 102.1 101.97830 101.0979  0.1217027  0.88041793
## Mar 2018 101.2 100.80670 100.9083  0.3933017 -0.10164685
##
## Forecasts:
##                y_f     sa_f      t_f         s_f          i_f
## Apr 2018 101.8030 100.8442 100.8714  0.9587153 -0.027176702
## May 2018 102.3456 100.6397 101.0182  1.7059430 -0.378560946
## Jun 2018 102.3288 101.2726 101.3284  1.0561489 -0.055725767
## Jul 2018 102.0091 101.7807 101.7358  0.2283849  0.044981509
```

CVS-CJO : exemples (7/8)

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



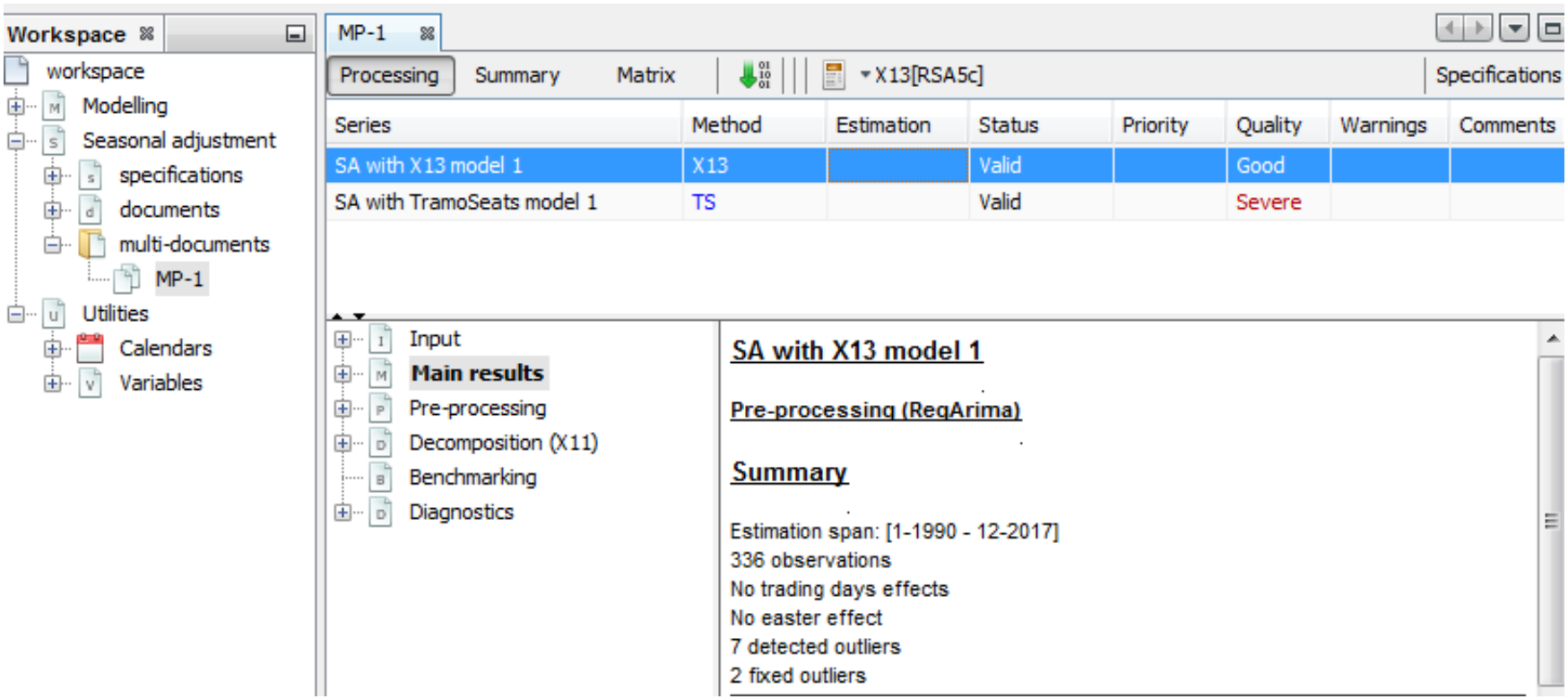
CVS-CJO : exemples (8/8)

```
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         25.022
##  Seasonal       7.085
##  Irregular      4.953
##  TD & Hol.      0.000
##  Others        64.807
##  Total        101.868
##
##  Combined test in the entire series
##  Non parametric tests for stable seasonality
##                                                                  P.value
##    Kruskall-Wallis test                                                0
##    Test for the presence of seasonality assuming stability             0
##    Evolutive seasonality test                                          0
##
##  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")
```

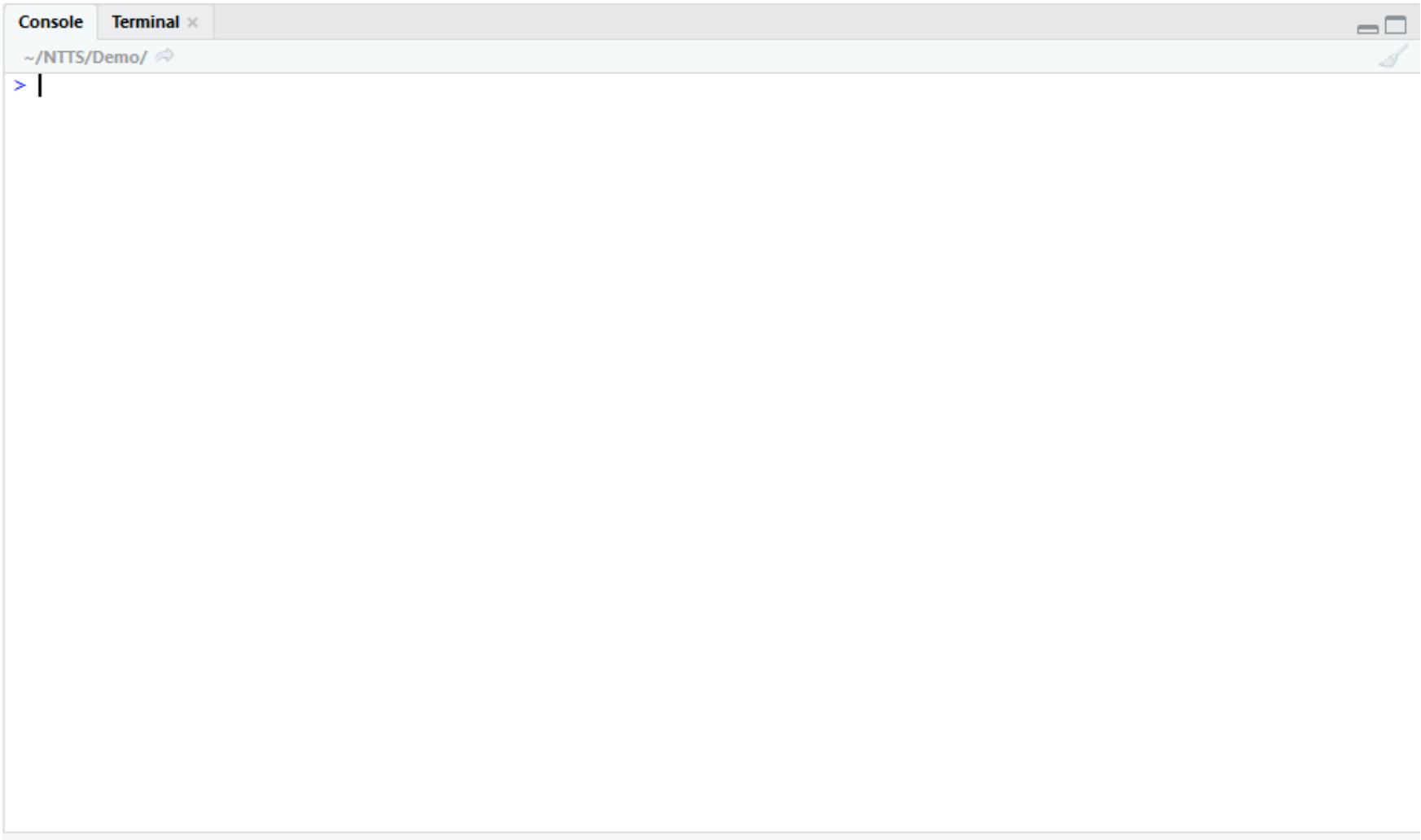


Importer un workspace (1/3)

```
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
## 1997  60.8  60.5  61.3  62.4  62.3  62.0  61.8  61.5  61.1  61.5  62.4  63.2
## 1998  63.7  63.8  65.7  66.0  66.3  67.1  65.7  64.9  64.3  64.0  65.1  66.8
## 1999  66.1  65.2  64.5  66.5  66.7  65.8  65.7  66.1  65.4  64.9  64.5  64.6
## 2000  64.7  65.3  66.7  67.4  68.1  69.8  69.3  68.9  68.5  69.6  69.7  71.0
## 2001  69.5  68.8  70.0  70.8  71.9  71.6  71.8  71.1  70.9  71.0  71.6  72.6
## 2002  71.3  71.3  71.5  72.7  72.5  73.7  75.1  73.1  72.0  72.3  73.3  73.5
## 2003  73.4  73.7  73.8  74.2  74.3  74.0  74.1  73.2  72.9  73.8  74.1  74.1
## 2004  74.3  74.0  73.4  73.9  74.4  74.6  74.8  74.0  74.1  74.2  75.6  76.0
## 2005  76.1  76.2  77.9  77.9  78.1  78.7  79.5  79.3  79.5  79.7  79.5  78.9
## 2006  79.7  80.7  81.1  81.9  82.7  82.0  80.6  79.9  82.0  81.3  81.8  83.0
## 2007  83.3  82.1  81.9  81.5  81.8  83.3  82.1  80.7  81.7  83.3  84.4  83.2
## 2008  86.2  86.1  86.7  87.2  87.9  88.4  90.1  91.6  90.5  91.2  91.4  90.7
## 2009  89.9  88.0  87.3  88.8  90.4  88.4  90.7  89.7  89.8  89.9  89.8  89.8
## 2010  89.7  89.4  90.0  91.0  91.6  92.6  91.2  90.6  91.2  92.1  93.0  93.4
## 2011  94.3  92.3  92.0  93.2  93.8  94.2  94.3  94.2  93.8  94.1  94.4  95.1
## 2012  99.8  98.1  98.8  99.2 100.5 100.6  99.9  99.6 100.7 100.8 101.9 101.6
## 2013 101.5 101.6 101.4 102.3 102.4 101.3 102.3 101.2 100.9  99.7  99.1  99.7
```

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(ipch_benin, x13_usr_spec)
# To get the available outputs:
tail(get_dictionary(jx13_mod), 2)
```

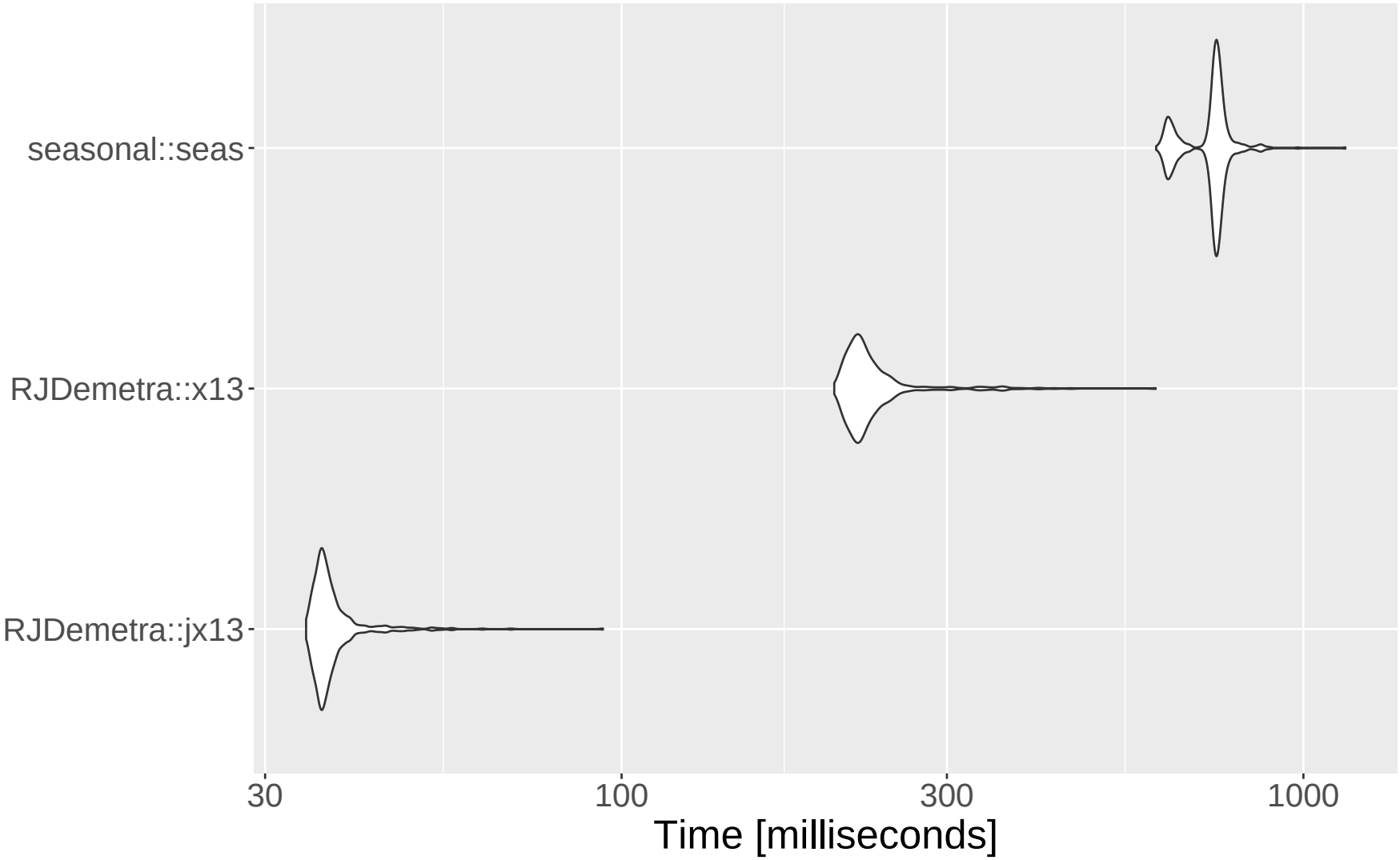
```
## [1] "diagnostics.msr-global" "diagnostics.msr(*)"
```

```
# To get an indicator:
get_indicators(jx13_mod, "diagnostics.ic-ratio")
```

```
## $`diagnostics.ic-ratio`
## [1] 2.020626
```

```
# 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

Travaux pratiques 

Maintenant à vous de jouer !

Documents sous : https://github.com/AQLT/BCEAO_2020

Objectifs du TP :

- Prendre en main `rjwsacruncher` et mettre à jour son workspace
- Prendre en main `RJDemetra` : faire une désaisonnalisation sous R, changer la spécification, exporter et importer un workspace.