PERFECTIONNEMENT

Le package RJDemetra

Anna Smyk et Tanguy Barthélémy
Division Recueil et Traitement de l'Information
Département des Méthodes Statistiques

Sommaire

RJDemetra

RJDemetra est un package qui permet :

- d'utiliser les algorithmes de JDemetra +
- d'avoir accès à l'output (séries et diagnostics). . . ...directement en R

: https://github.com/rjdverse/rjdemetra

- disponible sur le CRAN
- fonctions entièrement documentées

Pour l'installer :

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

Fonctionnalités

- Utiliser : RegARIMA, TRAMO-SEATS et X-13-ARIMA :
  - spécifications prédéfinies et personnalisées
  - graphiques, édition des paramètres et diagnostics
- Manipulation de workspaces JD+ :
  - Import de workspaces avec les paramètres de l'ajustement saisonnier
  - Export des modèles créés avec RJDemetra en xml, lisibles par l'interface JDemetra+ ou le cruncher
- Contient une base de données : les IPI dans l'industrie manufacturière dans l'UE

Estimation d'une cvs-cjo

```
library("RJDemetra")
# serie brute 1
ipi_fr <- ipi_c_eu[, "FR"]
# ou serie brute 2 : creation d'un objet TS à partir d'un data fr
# serie_brute <- ts(ipi[, "RF3030"], frequency=12, start=c(1990,
model_sa <- x13(ipi_fr, spec = "RSA5c")
```

L'objet crée `model_SA` contient : - les séries (brute et estimées) - les paramètres - les diagnostics

(noms identiques à ceux de l'interface)

Structure de l'objet model_SA

Un objet `model_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
```

Personnalisation des paramètres

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

```
# modification d'une specification
x13_usr_spec <- x13_spec(
    spec = "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"
)
# re-estimation avec la nouvelle spec
x13_mod <- x13(ipi_fr, x13_usr_spec)
ts_mod <- tramoseats(ipi_fr, "RSAfull")
```

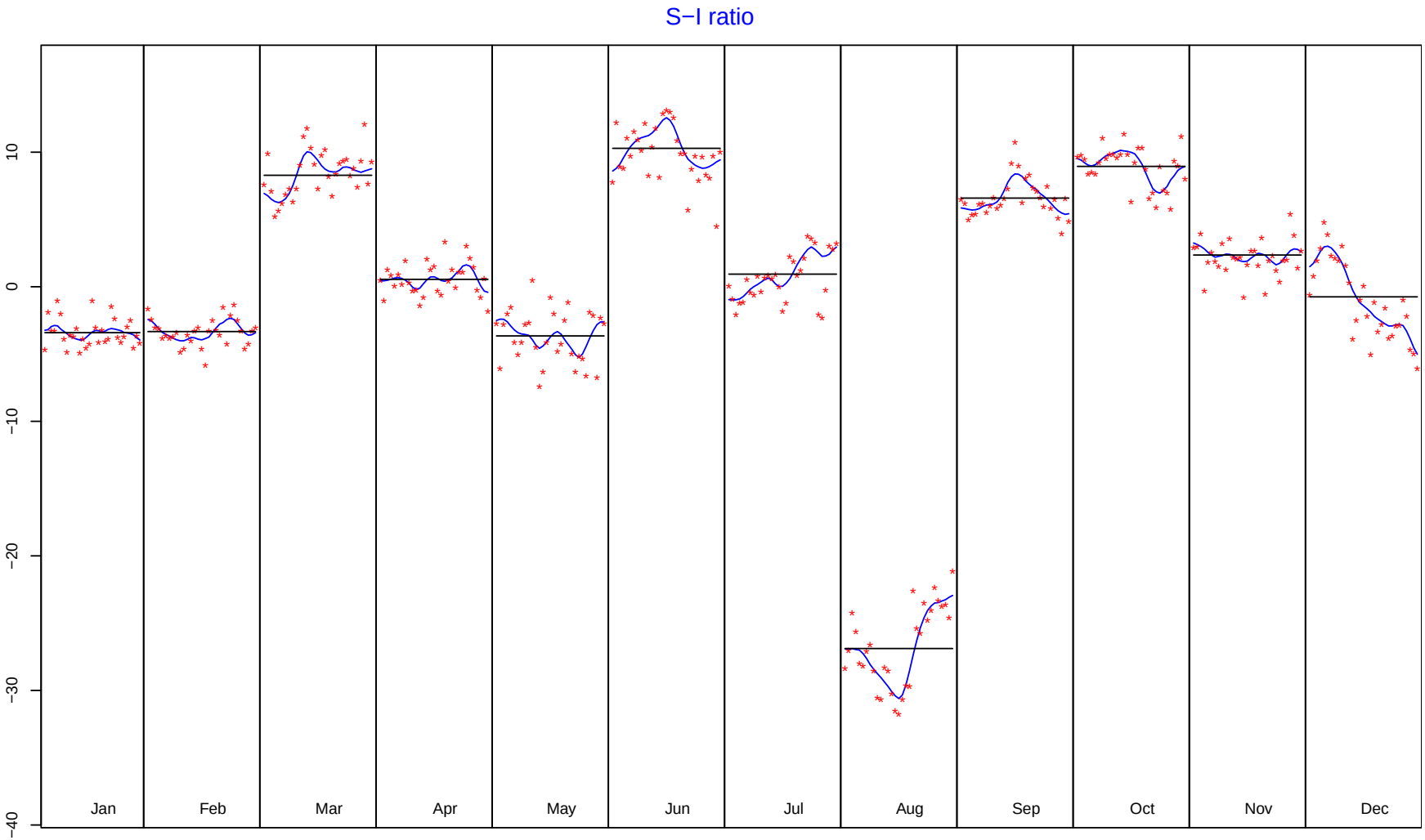
Affichage diagnostics decomposition

```
x13_mod$decomposition
```

```
## Monitoring and Quality Assessment Statistics:
##        M stats
## M(1)     0.151
## M(2)     0.097
## M(3)     1.206
## M(4)     0.558
## M(5)     1.041
## M(6)     0.037
## M(7)     0.082
## M(8)     0.242
## M(9)     0.062
## M(10)    0.267
## M(11)    0.252
## Q        0.366
## Q-M2     0.399
##
## Final filters:
## Seasonal filter:  3x5
## Trend filter:  13 terms Henderson moving average
```

Plot SI-ratios

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



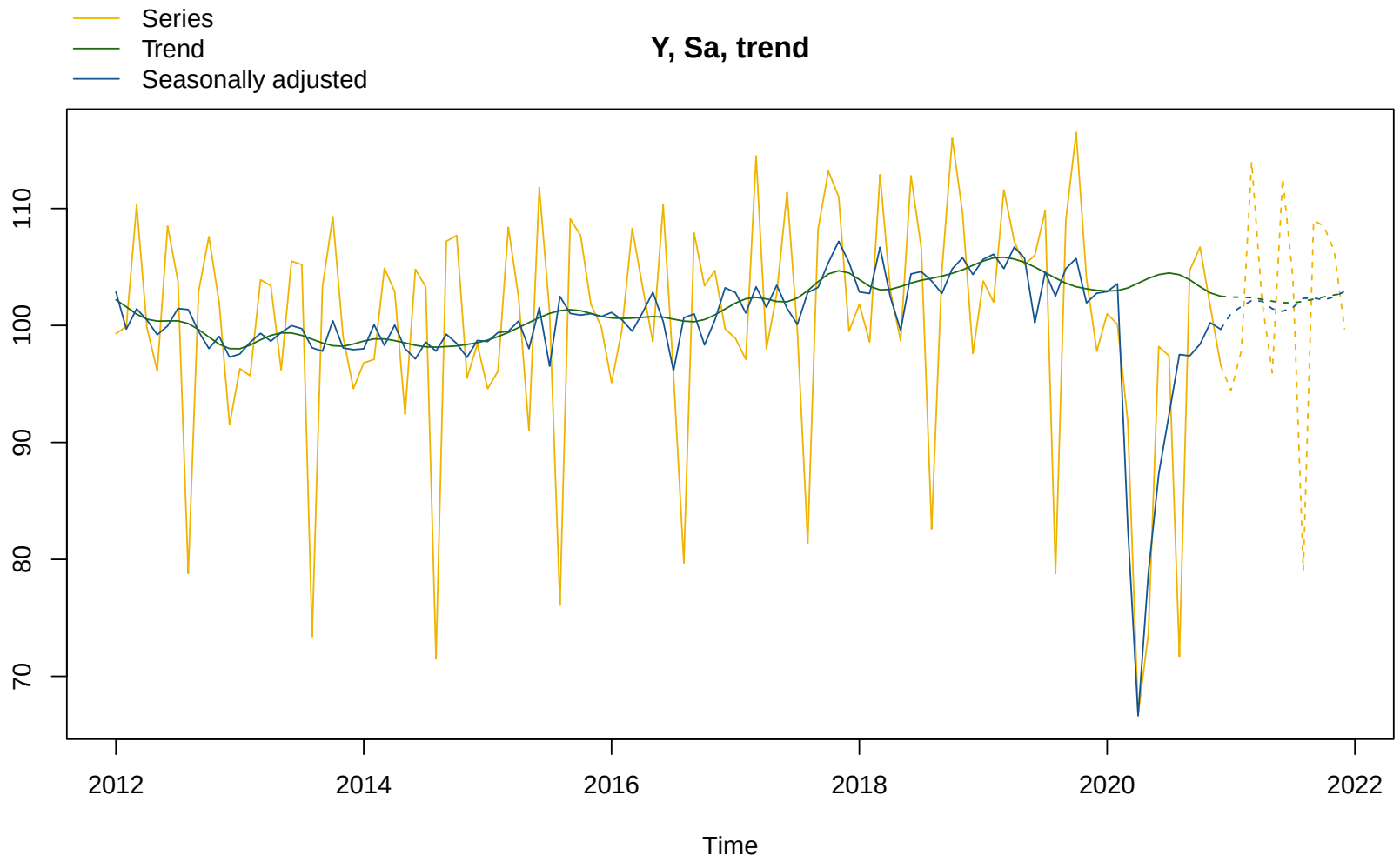
CVS-CJO : exemple (6/8)

```
x13_mod$final
```

```
## Last observed values
##                y        sa        t             s           i
## Jan 2020 101.0 102.89447 102.9447  -1.89446776  -0.0502488
## Feb 2020 100.1 103.56224 102.9860  -3.46224124   0.5762734
## Mar 2020  91.8  82.81896 103.2071   8.98103618 -20.3881828
## Apr 2020  66.7  66.62390 103.6164   0.07610348 -36.9925073
## May 2020  73.7  78.88976 104.0255  -5.18976181 -25.1357871
## Jun 2020  98.2  87.30845 104.3450  10.89154932 -17.0365408
## Jul 2020  97.4  92.39390 104.4861   5.00609785 -12.0921816
## Aug 2020  71.7  97.51560 104.3380 -25.81559971  -6.8224392
## Sep 2020 104.7  97.40102 103.9044   7.29897634  -6.5033820
## Oct 2020 106.7  98.39408 103.3109   8.30592464  -4.9168409
## Nov 2020 101.6 100.23574 102.7824   1.36426365  -2.5467131
## Dec 2020  96.6  99.67219 102.4984  -3.07218537  -2.8261840
##
## Forecasts:
##                 y_f     sa_f      t_f           s_f         i_f
## Jan 2021   94.41766 101.0272 102.4220  -6.60952495 -1.39481900
## Feb 2021   97.82331 101.6172 102.4196  -3.79385040 -0.80247216
## Mar 2021 114.01485 102.1273 102.3712  11.88751670 -0.24388469
## Apr 2021 102.04691 102.0672 102.2273  -0.02033583 -0.16002624
                                                              7
## Jun 2021 112.81492 101.2188 101.9501  11.39833187 -0.7334891
```

Main Plot raw-sa-trend

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



RegARIMA : exemple (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)     0.05291      0.108
## Phi(2)     0.18672      0.074
## Theta(1)  -0.52137      0.103
## BTheta(1) -0.66132      0.042
##
##              Estimate Std. Error
## Week days      0.6927      0.031
## Leap year      2.0903      0.694
## Easter [1]    -2.5476      0.442
```

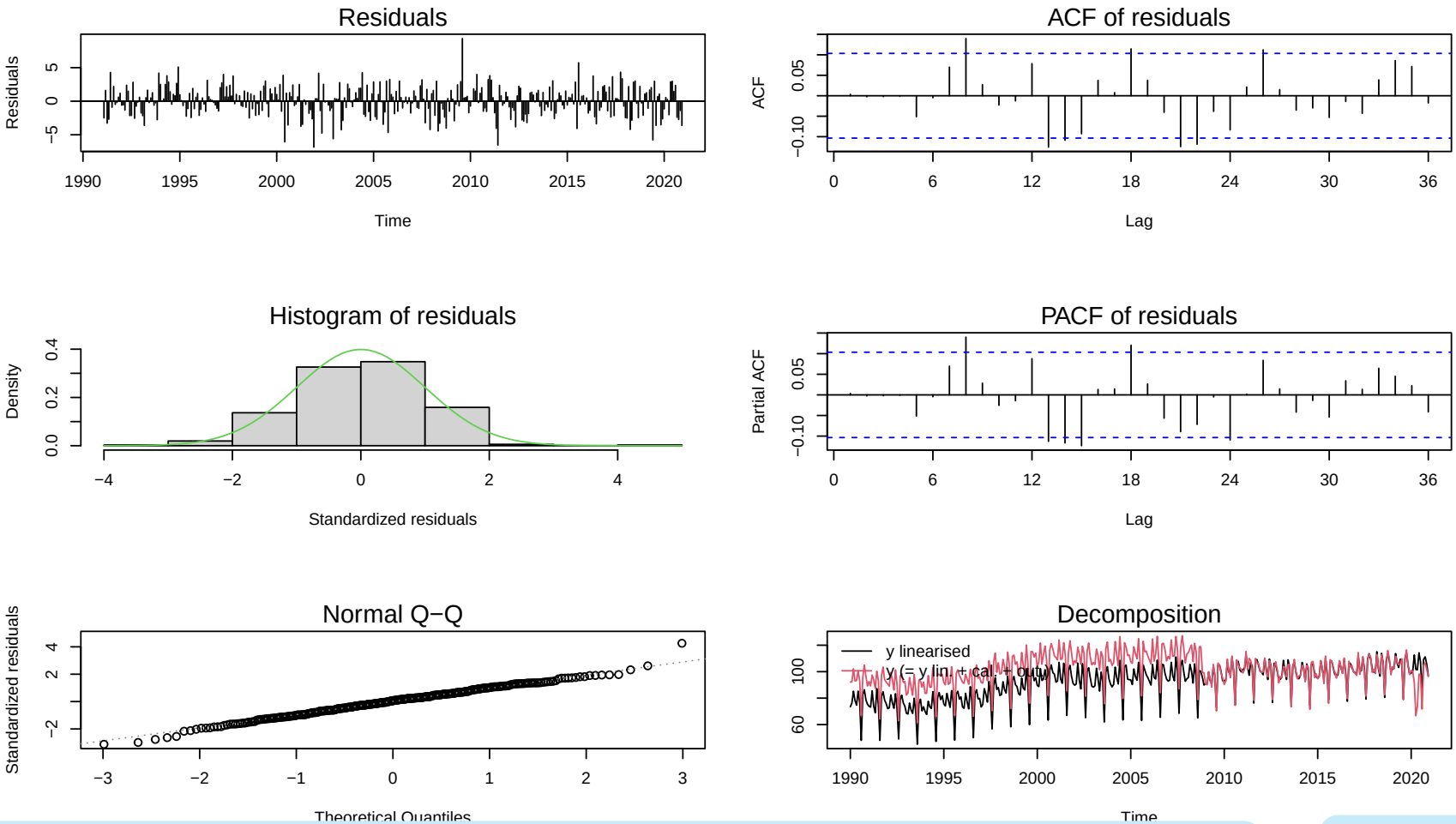
RegARIMA : exemple (2/4)

```
summary(regarima_model)
```

```
## y = regression model + arima (2, 1, 1, 0, 1, 1)
##
## Model: RegARIMA - X13
## Estimation span: from 1-1990 to 12-2020
## Log-transformation: no
## Regression model: no mean, trading days effect(2), leap year effect, Easter e
##
## Coefficients:
## ARIMA:
##           Estimate Std. Error  T-stat Pr(>|t|)
## Phi(1)     0.05291    0.10751   0.492    0.623
## Phi(2)     0.18672    0.07397   2.524    0.012 *
## Theta(1)  -0.52137    0.10270  -5.076 6.19e-07 ***
## BTheta(1) -0.66132    0.04222 -15.665  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Regression model:
##               Estimate Std. Error  T-stat Pr(>|t|)
## Week days      0.69265    0.03143  22.039  < 2e-16 ***
## Leap year      2.09030    0.69411   3.011  0.00278 **
## LC (4-2020)  -35.64811    2.09186 -17.041  < 2e-16 ***
```

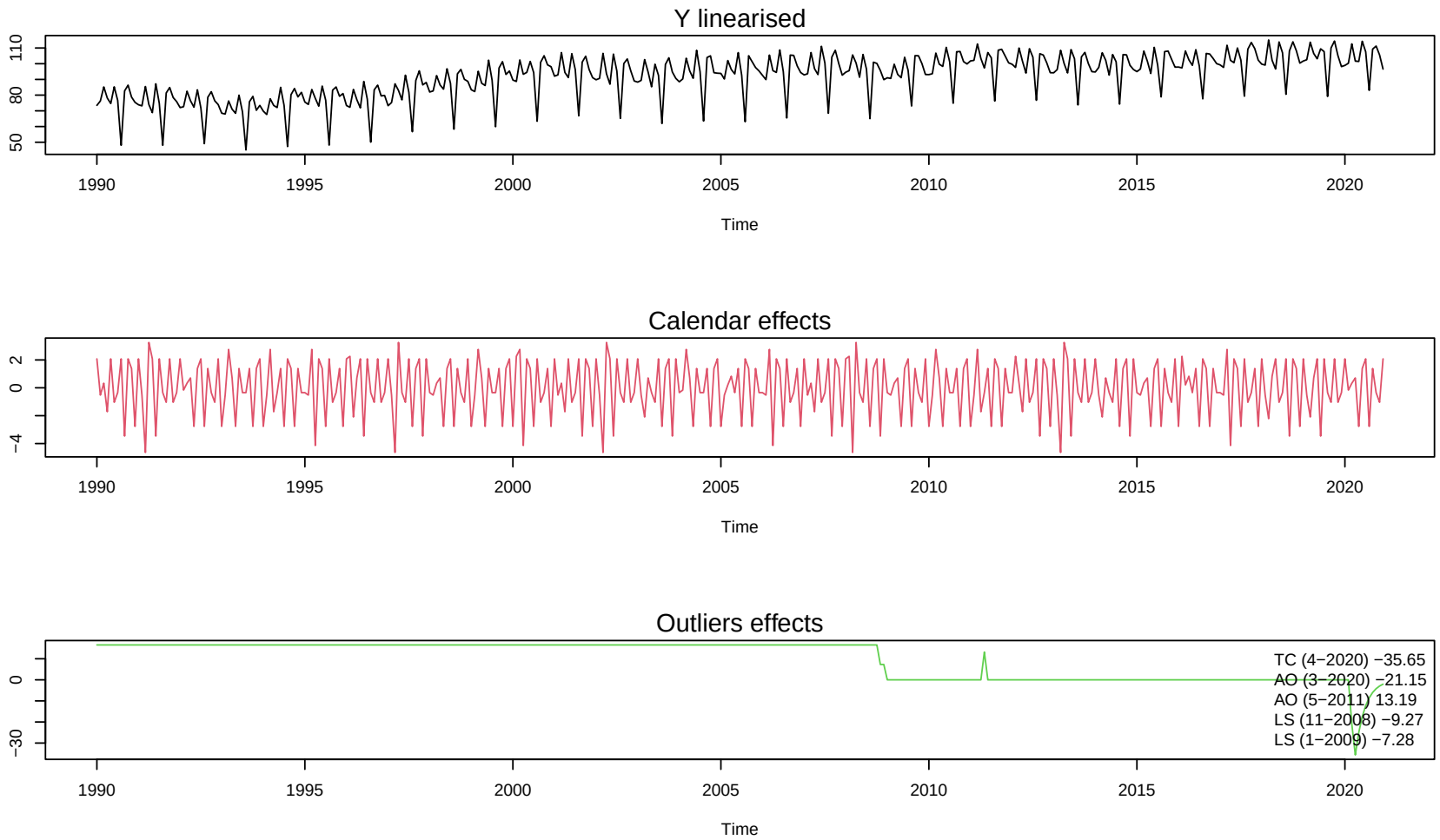
RegARIMA : exemple (3/4)

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



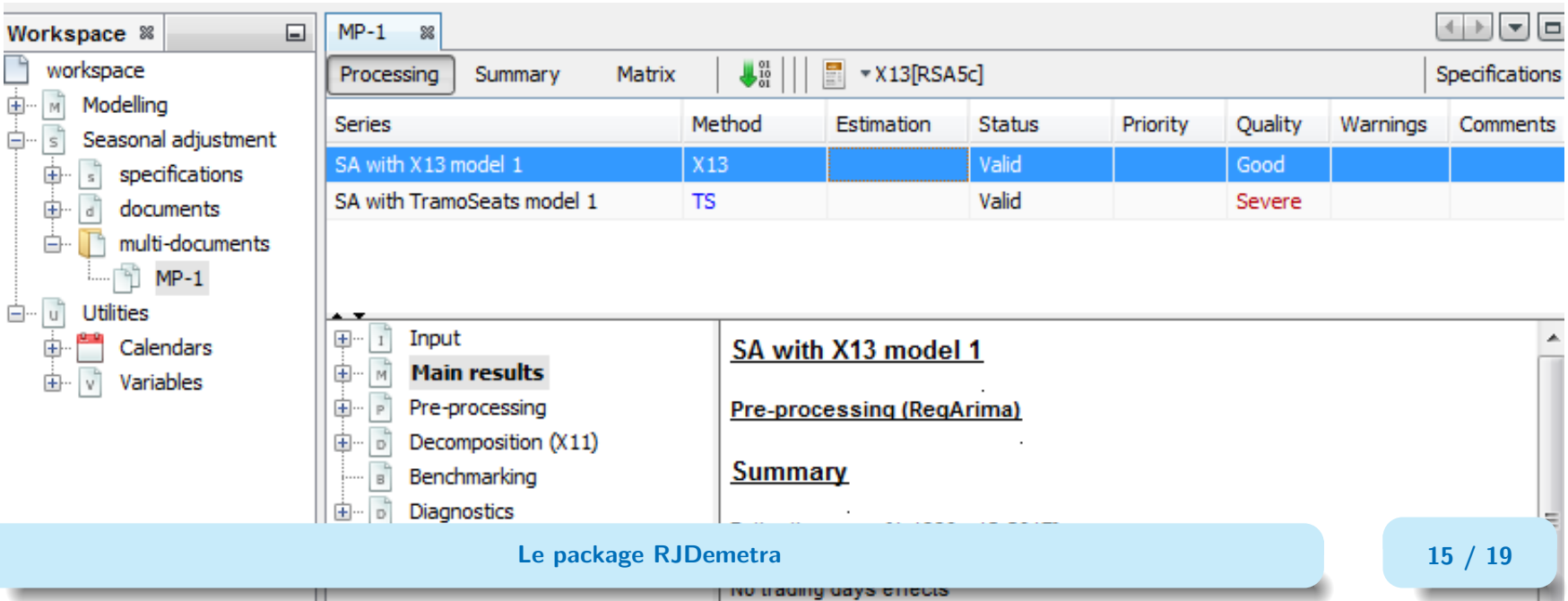
RegARIMA : exemple (4/4)

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



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

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

```
## Multiprocessing 1 on 1:
##   |
```

```
# 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)
```

```
## Multiprocessing 1 on 1:
##   |
```

Analyser l'output de JDemetra + sous R

- Possibilité de comparaison de différentes versions d'une même série désaisonnalisée avec plusieurs jeux de paramètres, stockés dans différents workspaces... sans créer d'output... au fil de l'analyse...

Exemple détaillé fourni :
programme Recuperer_UNE_serie_dans_N_WS_avec_RJD.R

Les autres packages s'appuyant sur RJDemetra

- **rjdqa** : qa = "quality assessment", produit un "dashboard" détaillé par série
- **rjdmarkdown** : édition de paramètres et diagnostics
- **ggdemetra** : graphiques enrichis
- **rjdworkspace** : changement de specs, fusion de workspaces (communication à venir)

Manque à la panoplie en version 2 : possibilité de mettre à jour une serie lors de l'arrivée d'un nouveau point brut : refresh policies (disponibles en v3)

exemple détaillés d'utilisation

- Utiliser les fonctionnalités de JDemetra +, sans ouvrir l'interface graphique ni le cruncher

Voir code fourni : CVS_en_R_avec_RJD.R

- Accéder à l'output créé par JDemetra+ et faire des comparaisons

Voir code : programme Recuperer_UNE_serie_dans_N_WS_avec_RJD.R

- Document de travail "R Tools for JDemetra+" disponible sur site insee.fr et dans biblio