

DÉSAISONNALISER UNE SÉRIE TEMPORELLE



2 - R et JDemetra+

ALAIN QUARTIER-LA-TENTE

Sommaire

1. Lancer JDemetra+ depuis R

1.1 Current status

1.2 RegARIMA : exemples

1.3 CVS-CJO : exemples

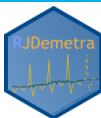
1.4 Manipuler des workspaces

2. Réduction du temps de calcul

3. Utilisation de RJDemetra pour améliorer la production

4. Lancement du JWSACruncher depuis R

5. R et JDemetra+ 3.0



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 -> voir manuel d'installation

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 : 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)	0.05291	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 : 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 12-2020
```

```
Log-transformation: no
```

```
Regression model: no mean, trading days effect(2), leap year effect, Easter effect
```

```
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 ***

```
---
```

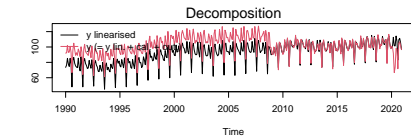
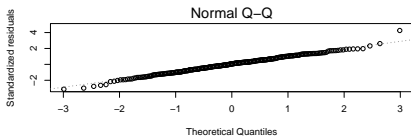
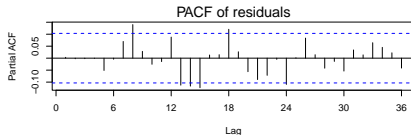
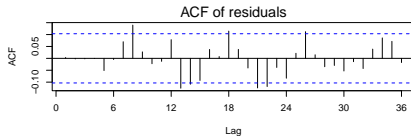
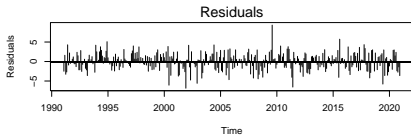
```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
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 **

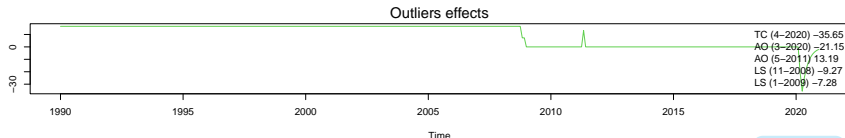
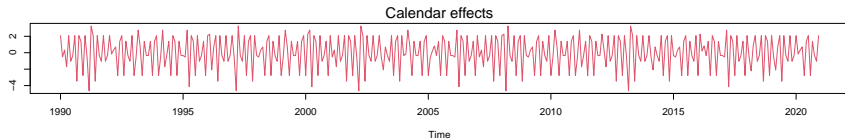
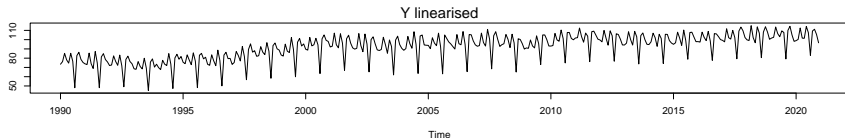
RegARIMA : exemples (3/4)

```
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 objet 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", "A0"),
                        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)    0.151
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
```

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

```
ts_mod$decomposition
```

Model

AR : $1 + 0.403230 B + 0.288342 B^2$

D : $1 - B - B^{12} + B^{13}$

MA : $1 - 0.664088 B^{12}$

SA

AR : $1 + 0.403230 B + 0.288342 B^2$

D : $1 - 2.000000 B + B^2$

MA : $1 - 0.970348 B + 0.005940 B^2 - 0.005813 B^3 + 0.003576 B^4$

Innovation variance: 0.7043507

Trend

D : $1 - 2.000000 B + B^2$

MA : $1 + 0.033519 B - 0.966481 B^2$

Innovation variance: 0.06093642

CVS-CJO : exemples (5/8)

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

S-I ratio



CVS-CJO : exemples (6/8)

x13_mod\$final

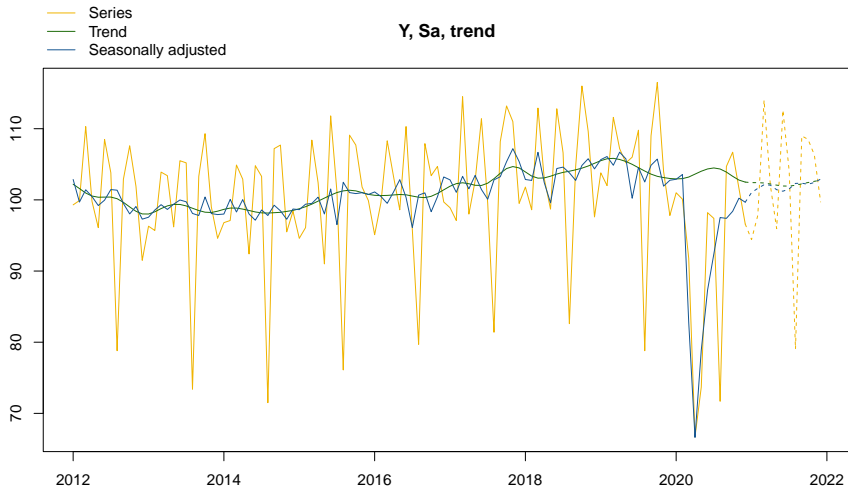
Last observed values

	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:

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	1.625
Seasonal	41.918
Irregular	0.727
TD & Hol.	1.851
Others	55.678
Total	101.800

Combined test in the entire series

Non parametric tests for stable seasonality

Kruskall-Wallis test

P.value

0.00

Exporter un workspace

```
jws <- new_workspace()
new_multiprocessing(jws, name = "MP-1")
add_sa_item(jws, multiprocessing = "MP-1",
            sa_obj = x13_mod, name = "SA with X13 model 1 ")
add_sa_item(jws, multiprocessing = "MP-1",
            sa_obj = ts_mod, name = "SA with TramoSeats model 1")
save_workspace(jws, "workspace.xml")
```

The screenshot displays the JDemetra+ workspace 'MP-1'. On the left, a tree view shows the workspace structure with folders for 'Modelling', 'Seasonal adjustment', 'specifications', 'documents', 'multi-documents', 'MP-1', 'Utilities', 'Calendars', and 'Variables'. The main window is divided into several panes:

- Processing Summary Table:** A table with columns: Series, Method, Estimation, Status, Priority, Quality, Warnings, and Comments.

Series	Method	Estimation	Status	Priority	Quality	Warnings	Comments
SA with X13 model 1	X13		Valid		Good		
SA with TramoSeats model 1	TS		Valid		Severe		
- Series Details:** A detailed view for the selected series 'SA with X13 model 1', showing:
 - Input**
 - Main results**
 - Pre-processing (ReqArima)**
 - Summary:** Estimation span: [1-1990 - 12-2017], 336 observations, No trading days effects, No season effect.

Importer un workspace (1/3)

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

```
$`MP-1`
```

```
$`MP-1`$`SA with X13 model 1 `
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1990	92.1	92.3	102.1	93.0	93.3	100.8	92.9	66.7	95.8	105.0	96.7	89.2
1991	92.5	89.2	97.4	93.8	87.5	100.3	93.4	64.3	96.9	103.5	94.0	92.1
1992	90.7	89.0	99.4	93.7	86.1	101.3	90.4	62.9	96.6	98.4	91.9	92.6
1993	82.3	84.0	95.6	88.3	82.2	97.9	85.5	61.3	93.7	93.0	88.3	92.1
1994	83.6	83.7	97.0	88.3	88.3	102.9	87.3	65.9	98.2	98.0	96.8	98.0
1995	91.8	90.1	102.9	90.4	91.6	103.7	90.6	66.8	98.7	101.4	97.2	94.8
1996	92.0	91.1	98.1	94.3	90.5	101.8	96.1	66.3	98.9	105.0	95.0	96.0
1997	91.9	91.3	99.1	102.8	93.2	108.2	100.4	70.5	107.3	114.1	99.6	106.7
1998	98.2	98.7	109.3	103.7	97.6	114.7	106.1	72.1	111.5	112.6	105.6	107.4
1999	97.2	98.3	114.5	104.8	99.9	120.2	105.7	76.1	115.2	115.1	111.1	114.0
2000	103.4	107.5	121.7	105.7	113.1	119.4	108.1	82.0	116.4	121.3	117.2	111.9
2001	110.7	108.9	124.0	109.3	109.8	121.9	112.4	85.5	114.1	123.4	114.2	104.9
2002	108.4	106.7	118.5	113.4	105.6	119.2	113.9	81.4	115.6	121.7	111.0	105.2
2003	106.9	105.4	117.1	112.0	101.5	115.2	111.2	75.7	117.5	122.4	107.8	109.3
2004	104.7	106.7	122.8	112.7	104.5	126.5	111.1	79.7	121.9	118.8	112.2	112.6
2005	107.6	106.3	118.8	113.7	109.7	125.0	106.4	81.7	123.0	115.1	115.5	111.6
2006	108.8	105.9	124.8	108.0	113.1	126.7	108.7	84.1	121.0	121.5	116.6	108.2

Importer un workspace (2/3)

Note : animation visible sur Adobe Reader uniquement

Importer un workspace (3/3)

```
compute(jws) # Important to get the Sa model
models <- get_model(jws) # A progress bar is printed by default
```

Multiprocessing 1 on 1:

```
|
|                                     | 0%
|=====                             | 50%
|=====                             | 100%
```

```
# To extract only one model
jmp <- get_object(jws, 1)
count(jmp)
```

```
[1] 2
```

```
jsa2 <- get_object(jmp, 2)
get_name(jsa2)
```

```
[1] "SA with TramoSeats model 1"
```

```
mod <- get_model(jws, jsa2)
```

Sommaire

1. Lancer JDemetra+ depuis R

2. Réduction du temps de calcul

2.1 Manipulation des objets Java

2.2 Benchmarking

3. Utilisation de RJDemetra pour améliorer la production

4. Lancement du JWSACruncher depuis R

5. R et JDemetra+ 3.0

Manipuler des objets ☹️ (1/2)

Les fonctions peuvent être assez consommatrices en temps de calcul... surtout si l'on n'a besoin que d'un seul paramètre

➡️ “Manipuler” modèles Java : `jx13`, `jtramoseats`, `jregarima`, `jregarima_x13`, `jregarima_tramoseats` et `get_jmodel`

```
jx13_mod <- jx13(ipi_fr, x13_usr_spec)
# To get the available outputs:
tail(get_dictionary(jx13_mod))
```

```
[1] "residuals.independence.value" "residuals.independence"
[3] "residuals.tdpeaks.value"     "residuals.tdpeaks"
[5] "residuals.seaspeaks.value"   "residuals.seaspeaks"
```

Manipuler des objets (2/2)

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

$`diagnostics.td-res-all`
[1] 0.1896922 0.9796182
attr(,"description")
[1] "F with 6 degrees of freedom in the nominator and 353 degrees of freedom in t

$`diagnostics.ic-ratio`
[1] 5.050485

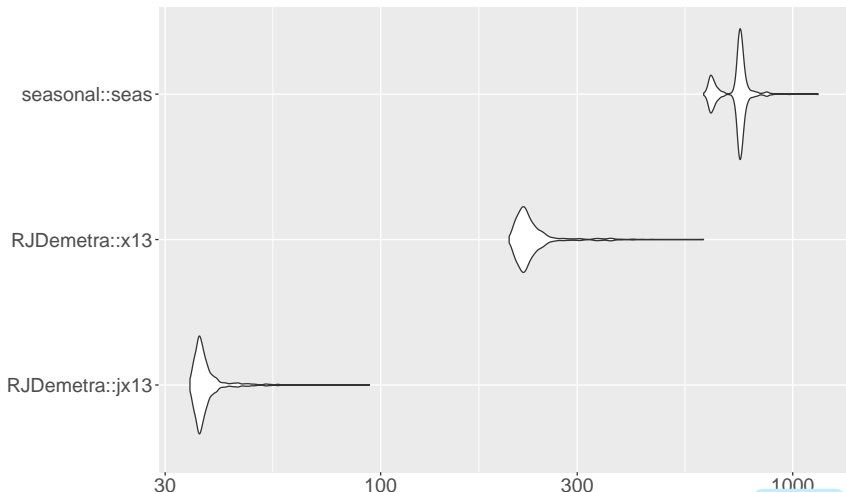
# To get the previous R output
x13_mod <- jSA2R(jx13_mod)
```

➔ L'output peut être personnalisé

⚠ Pas d'erreur renvoyé par `jx13()` avec une "mauvaise" SA (preliminary check...) and `get_indicators()` renvoie objet NULL

Benchmarking with X-13 on French IPI

R version 4.2.0 (2022-04-22), x86_64-apple-darwin17.0, Windows 7 x64 (build 7601)
Service Pack 1




Sommaire

1. Lancer JDemetra+ depuis R
2. Réduction du temps de calcul
- 3. Utilisation de RJDemetra pour améliorer la production**
 - 3.1 Autour de RJDemetra
4. Lancement du JWSACruncher depuis R
5. R et JDemetra+ 3.0

Exemples d'utilisation de RJDemetra

- rjdqa : package pour aider à évaluer la qualité de la désaisonnalisation (tableau de bord)

 <https://github.com/AQLT/rjdqa>

- ggdemetra : intégrer la désaisonnalisation à ggplot2

 <https://github.com/AQLT/ggdemetra>

- rjdmarkdown : faciliter les rapports automatiques avec rmarkdown

 <https://github.com/AQLT/rjdmarkdown>

- rjdworkspace : manipuler les workspaces

 <https://github.com/InseeFrLab/rjdworkspace>

- persephone (expérimental) : faciliter la production de séries CVS-CJO au sein de l'institut (graphiques interactifs, tableaux de bord...)

 <https://github.com/statistikat/persephone>

Sommaire

1. Lancer JDemetra+ depuis R
2. Réduction du temps de calcul
3. Utilisation de RJDemetra pour améliorer la production
- 4. Lancement du JWSACruncher depuis R**
5. R et JDemetra+ 3.0

Le JWSACruncher

Objectifs du JWSACruncher : 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 JWSACruncher
<https://github.com/jdemetra/jwsacruncher/releases>.
- l'aide associée au JWSACruncher
<https://github.com/jdemetra/jwsacruncher/wiki>.

Le JWSACruncher

Pour lancer le JWSACruncher de JDemetra+ il faut :

- le JWSACruncher ;
- 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+.

Sur le CRAN il y a le package `rjwsacruncher`
(<https://github.com/AQLT/rjwsacruncher>) qui facilite son utilisation !

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 =
        "D:/jwsacruncher-2.2.0/jdemetra-cli-2.2.0/bin")
```

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:/workspace.xml",
                    # Pour ne pas renommer les noms des dossiers e
                    rename_multi_documents = FALSE,
                    policy = "lastoutliers")
```

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

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

Sommaire

1. Lancer JDemetra+ depuis R
2. Réduction du temps de calcul
3. Utilisation de RJDemetra pour améliorer la production
4. Lancement du JWSACruncher depuis R
5. R et JDemetra+ 3.0

R et JDemetra+ 3.0 (1)

La version 3 de JDemetra+ est accompagnée de nombreux packages R uniquement disponible sur  github.com/rjdemetra :

- `rjd3toolkit` : boîte à outils (créations de régresseurs, de calendriers, tests, etc.)
- `rjd3x13` et `rjd3tramoseats` : méthodes X-13 et TRAMO-SEATS.
- `rjdemetra3` : manipulation de workspaces
- `rjd3x11plus`, `rjd3st1` et `rjd3highfreq` : différentes méthodes pour la désaisonnalisation de séries hautes fréquences
Exemples : https://github.com/palatej/test_rjd3hf
- `rjd3filters` : manipulation de moyennes mobiles
- `rjd3sts` : interface simple autour des modèles espace-état et des *structural time series*
Exemples : https://github.com/palatej/test_rjd3sts

R et JDemetra+ 3.0 (2)

- `rjd3bench` : *Benchmarking* (uni et multivarié) et désagrégation temporelle
Exemples : https://github.com/palatej/test_rjd3bench
- `rjd3revisions` : analyse des révisions
- `ggdemetra3` : version 3 de `ggdemetra`

Bibliographie

-  Alain Quartier-la-Tente, Anna Michalek, Jean Palate and Raf Baeyens (2021). *RJDemetra: Interface to 'JDemetra+' Seasonal Adjustment Software*. <https://github.com/jdemetra/RJDemetra>
-  Alain Quartier-la-Tente (2021). *rjdworkspace: Manipulation of JDemetra+ Workspaces*. <https://github.com/InseeFrLab/rjdworkspace>.
-  Alain Quartier-la-Tente. *rjdqa: Quality Assessment for Seasonal Adjustment*. <https://github.com/AQLT/rjdqa>.
-  Alain Quartier-la-Tente (2020). *rjdmarkdown: 'rmarkdown' Extension for Formatted 'RJDemetra' Outputs*. R package version 0.2.0. <https://github.com/AQLT/rjdmarkdown>.
-  Alain Quartier-la-Tente. *ggdemetra: 'ggplot2' Extension for Seasonal and Trading Day Adjustment with 'RJDemetra'*. <https://github.com/AQLT/ggdemetra>.
-  Alain Quartier-la-Tente (2019). *rjwsacruncher: Interface to the 'JWSACruncher' of 'JDemetra+'*. <https://github.com/AQLT/rjwsacruncher>
-  Anna Smyk, Alice Tchang (2021). *R Tools for JDemetra+, Seasonal adjustment made easier*. Insee, Document de travail n° M2021/01. <https://www.insee.fr/fr/statistiques/5019786>.