

RENCONTRES R 2023



Institut national de la statistique
et des études économiques

Mesurer pour comprendre

Manipuler les moyennes mobiles avec R et JDemetra+

ALAIN QUARTIER-LA-TENTE
21 au 23 juin 2023
INSEE, LEMNA

Contents

1. Introduction


2. rjd3filters et les moyennes mobiles

3. Autres packages

4. Conclusion


JDemetra+ ???

JDemetra+ :


- logiciel officiellement recommandé par Eurostat et la BCE pour la désaisonnalisation
- un package  associé : RJDemetra
- Nombreuses extensions qui permettent de traiter d'autres problèmes de séries temporelles : benchmarking, nowcasting, DFM, etc.

JDemetra+ ???

JDemetra+ :

- logiciel officiellement recommandé par Eurostat et la BCE pour la désaisonnalisation
- un package  associé : RJDemetra
- Nombreuses extensions qui permettent de traiter d'autres problèmes de séries temporelles : benchmarking, nowcasting, DFM, etc.

JDemetra+ 3.0 :

- Refactorisation du code qui facilite l'accès au code via d'autres logiciels
- Utilisation des Protocol Buffers pour accélérer l'interaction avec Java (RProtoBuf)
- Développement $\simeq 10$ de package  autour de problèmes de séries temporelles
- Aujourd'hui : focus sur un package (traitement des moyennes mobiles) et rapide présentation des autres

Contents

1. Introduction

2. rjd3filters et les moyennes mobiles

2.1 Introduction aux moyennes mobiles

2.2 Existant sous

2.3 rjd3filters

3. Autres packages

4. Conclusion

Introduction

Moyennes mobiles (MM) omniprésentes dans les problèmes de séries temporelles :

$$M_{\theta}(X_t) = \sum_{k=-p}^{+f} \theta_k X_{t+k} = \left(\sum_{k=-p}^{+f} \theta_k B^{-k} \right) X_t \text{ avec } B^k = X_{t-k}$$

Introduction

Moyennes mobiles (MM) omniprésentes dans les problèmes de séries temporelles :

$$M_{\theta}(X_t) = \sum_{k=-p}^{+f} \theta_k X_{t+k} = \left(\sum_{k=-p}^{+f} \theta_k B^{-k} \right) X_t \text{ avec } B^k = X_{t-k}$$

Exemple :

- Lissage : moyennes mobiles simples (ex : COVID), LOESS (`ggplot2::geom_smooth()`), régressions locale, etc.

Introduction

Moyennes mobiles (MM) omniprésentes dans les problèmes de séries temporelles :

$$M_{\theta}(X_t) = \sum_{k=-p}^{+f} \theta_k X_{t+k} = \left(\sum_{k=-p}^{+f} \theta_k B^{-k} \right) X_t \text{ avec } B^k = X_{t-k}$$

Exemple :

- Lissage : moyennes mobiles simples (ex : COVID), LOESS (`ggplot2::geom_smooth()`), régressions locale, etc.
- Prévisions : ARIMA

Introduction

Moyennes mobiles (MM) omniprésentes dans les problèmes de séries temporelles :

$$M_{\theta}(X_t) = \sum_{k=-p}^{+f} \theta_k X_{t+k} = \left(\sum_{k=-p}^{+f} \theta_k B^{-k} \right) X_t \text{ avec } B^k = X_{t-k}$$

Exemple :

- Lissage : moyennes mobiles simples (ex : COVID), LOESS (`ggplot2::geom_smooth()`), régressions locale, etc.
- Prévisions : ARIMA
- Décomposition : désaisonnalisation (X-13ARIMA, STL), extraction de tendance-cycle (analyse du cycle des affaires)

Qu'est-ce qui existe sous ? (1)

Pour la manipulation de MM :

1. `stats::filter(., method= "recursive", sides = 2)` : MM symétriques (p impair)

$$y_t = f_1 x_{t+\lceil(p-1)/2\rceil} + \dots + f_p x_{t+\lceil(p-1)/2\rceil-(p-1)}$$

or `stats::filter(., method= "recursive", sides = 1)` : MM utilisées en temps-réel

$$y_t = f_1 x_t + \dots + f_p x_{t-(p-1)}$$

Qu'est-ce qui existe sous ? (1)

Pour la manipulation de MM :

1. `stats::filter(., method= "recursive", sides = 2)` : MM symétriques (p impair)

$$y_t = f_1 x_{t+\lceil(p-1)/2\rceil} + \dots + f_p x_{t+\lceil(p-1)/2\rceil-(p-1)}$$

or `stats::filter(., method= "recursive", sides = 1)` : MM utilisées en temps-réel

$$y_t = f_1 x_t + \dots + f_p x_{t-(p-1)}$$

➡ On peut ajouter des 0 pour généraliser mais on perd l'estimation des derniers points

Qu'est-ce qui existe sous ? (2)

Régression locale :

2. `KernSmooth::locpoly()` régression locale avec noyau gaussien
3. `locfit::locfit()` régression locale avec tricube, rectangulaire, triweight, triangulaire, epanechnikov, bisquare, gaussien
4. `stats::loess()` noyau tricube

Qu'est-ce qui existe sous ? (2)

Régression locale :



2. `KernSmooth::locpoly()` régression locale avec noyau gaussien
3. `locfit::locfit()` régression locale avec tricube, rectangulaire, triweight, triangulaire, epanechnikov, bisquare, gaussien
4. `stats::loess()` noyau tricube
5. Désaisonnalisation : `stats`, `seasonal`, `RJDemetra`, `x12` : difficile d'isoler les étapes et les différentes moyennes mobiles utilisées

Qu'est-ce qui existe sous R ? (2)

Régression locale :



2. `KernSmooth::locpoly()` régression locale avec noyau gaussien
 3. `locfit::locfit()` régression locale avec tricube, rectangulaire, triweight, triangulaire, epanechnikov, bisquare, gaussien
 4. `stats::loess()` noyau tricube
 5. Désaisonnalisation : `stats`, `seasonal`, `RJDemetra`, `x12` : difficile d'isoler les étapes et les différentes moyennes mobiles utilisées
-
- ➡ Aucune fonction pour manipuler facilement les MM et analyser leurs propriétés statistiques (gain, déphasage, etc.)
 - ➡ Aucune moyen de créer les moyennes mobiles classiques liées à la désaisonnalisation : Henderson, Musgrave, Macurves, etc.

rjd3filters (1)

rjd3filters : package  basé sur les librairies  de JDemetra+ 3.0 qui permet de :



- créer/combiner/appliquer facilement toutes les MM `moving_average()`

rjd3filters (1)

rjd3filters : package  basé sur les librairies  de JDemetra+ 3.0 qui permet de :



- créer/combiner/appliquer facilement toutes les MM `moving_average()`
- créer/combiner/appliquer facilement un ensemble de MM : une MM centrale (estimations finales) et des MM lorsque les données ne sont pas disponibles `finite_filters()`

rjd3filters (1)

rjd3filters : package  basé sur les librairies  de JDemetra+ 3.0 qui permet de :

- créer/combiner/appliquer facilement toutes les MM `moving_average()`
- créer/combiner/appliquer facilement un ensemble de MM : une MM centrale (estimations finales) et des MM lorsque les données ne sont pas disponibles `finite_filters()`
- étudier les propriétés des MM : courbe des coefficients (`plot_coef()`), gain (`plot_gain()`), déphasage (`plot_phase()`) et différentes statistiques (`diagnostic_matrix()`)

rjd3filters (1)

rjd3filters : package  basé sur les librairies  de JDemetra+ 3.0 qui permet de :

- créer/combiner/appliquer facilement toutes les MM `moving_average()`
- créer/combiner/appliquer facilement un ensemble de MM : une MM centrale (estimations finales) et des MM lorsque les données ne sont pas disponibles `finite_filters()`
- étudier les propriétés des MM : courbe des coefficients (`plot_coef()`), gain (`plot_gain()`), déphasage (`plot_phase()`) et différentes statistiques (`diagnostic_matrix()`)
- différentes méthodes de construction de MM pour l'extraction de la tendance-cycle et “personnaliser” X-11

rjd3filters (2)

Disponible sous  palatej/rjd3filters

```
devtools::install_github("palatej/rjd3toolkit")  
devtools::install_github("palatej/rjd3filters")
```

Création et manipulations moving_average() (1)

(Rappel : $B^i X_t = X_{t-p}$ et $F^i X_t = X_{t+p}$)

```
library(rjd3filters)
m1 = moving_average(rep(1,4), lags = -2)/4; m1 # MM simple d e4 termes
```

```
## [1] "0.2500 B^2 + 0.2500 B + 0.2500 + 0.2500 F"
```

```
m2 = moving_average(rep(1,3), lags = -1)/3; m2 # MM centrés 3 termes
```

```
## [1] "0.3333 B + 0.3333 + 0.3333 F"
```

```
m1 + m2
```

```
## [1] "0.2500 B^2 + 0.5833 B + 0.5833 + 0.5833 F"
```

```
m1 - m2
```

```
## [1] "0.2500 B^2 - 0.0833 B - 0.0833 - 0.0833 F"
```

```
m1 * m2
```

```
## [1] "0.0833 B^3 + 0.1667 B^2 + 0.2500 B + 0.2500 + 0.1667 F + 0.0833 F^2"
```

```
m1^2
```

Création et manipulations `moving_average()` (2)

```
## [1] "0.0625 B^4 + 0.1250 B^3 + 0.1875 B^2 + 0.2500 B + 0.1875 + 0.1250 F + 0.0625 F^2"
rev(m1)
```

```
## [1] "0.2500 B + 0.2500 + 0.2500 F + 0.2500 F^2"
```

```
library(rjd3filters)
```

```
m1 = moving_average(rep(1,4), lags = -2)/4; m1 # MM simple d e4 termes
```

```
## [1] "0.2500 B^2 + 0.2500 B + 0.2500 + 0.2500 F"
```

```
m2 = moving_average(rep(1,3), lags = -1)/3; m2 # MM centrés 3 termes
```

```
## [1] "0.3333 B + 0.3333 + 0.3333 F"
```

```
m1 + m2
```

```
## [1] "0.2500 B^2 + 0.5833 B + 0.5833 + 0.5833 F"
```

```
m1 - m2
```

```
## [1] "0.2500 B^2 - 0.0833 B - 0.0833 - 0.0833 F"
```

```
m1 * m2
```

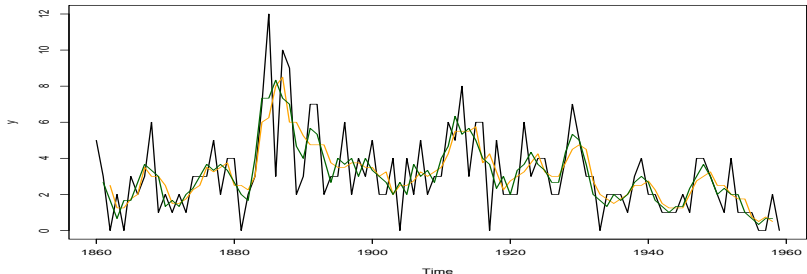
Création et manipulations moving_average() (3)

```
## [1] "0.0833 B^3 + 0.1667 B^2 + 0.2500 B + 0.2500 + 0.1667 F + 0.0833 F^2"  
m1^2
```

```
## [1] "0.0625 B^4 + 0.1250 B^3 + 0.1875 B^2 + 0.2500 B + 0.1875 + 0.1250 F + 0.  
rev(m1)
```

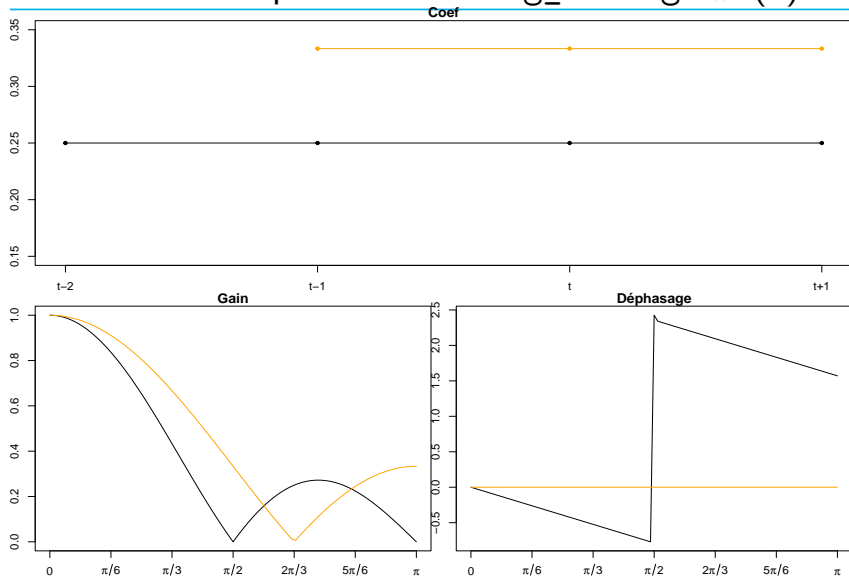
```
## [1] "0.2500 B + 0.2500 + 0.2500 F + 0.2500 F^2"  
y <- datasets::discoveries  
plot(y)  
lines(y * m1, col = "orange")  
lines(y * m2, col = "darkgreen")
```

Création et manipulations `moving_average()` (4)



```
par(mai = c(0.3, 0.3, 0.2, 0))  
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))  
plot_coef(m1, main = "Coef");plot_coef(m2, col = "orange", add = TRUE)  
plot_gain(m1, main = "Gain");plot_gain(m2, col = "orange", add = TRUE)  
plot_phase(m1, main = "Déphasage");plot_phase(m2, col = "orange", add = TRUE)
```

Création et manipulations `moving_average()` (5)



A quoi ça sert ?

- Dans les cas simples : inutile

A quoi ça sert ?

- Dans les cas simples : inutile
- Dans les cas plus compliqués (ex : désaisonnalisation, lissage) :
 - Aide à la formation (décomposition des étapes)
 - Aide à la recherche (modification de certaines étapes)
- Exemple : désaisonnalisation avec X-11 (combinaison de nombreuses moyennes mobiles)

Exemple X-11 (1)

```
y <- log(AirPassengers)
# On suppose  $Y = \text{Tendance-Cycle (TC)} + \text{Saisonnalité (S)} + \text{Irrégulier (I)}$ 
tc_1 <- M2X12 <- (simple_ma(12, - 6) + simple_ma(12, - 5))/2
si_1 <- 1 - tc_1
M3X3 <- macurves("S3x3")
M3X3 # MM sym pour l'estimation finale et MMA pour les est intermédiaires
```

```
##           q=2           q=1           q=0
## t-2 0.1111111 0.1111111 0.1851852
## t-1 0.2222222 0.2592593 0.4074074
## t    0.3333333 0.3703704 0.4074074
## t+1 0.2222222 0.2592593 0.0000000
## t+2 0.1111111 0.0000000 0.0000000
```

Exemple X-11 (2)

```
M3X3_s <- to_seasonal(M3X3, 12)
s_1 <- M3X3_s * si_1
s_1_norm <- M2X12 * s_1
s_1_norm <- impute_last_obs(s_1_norm, n = 6, nperiod = 1)
s_1_demean <- s_1 - s_1_norm
s_1_f <- impute_last_obs(s_1_demean, n = 6, nperiod = 12)
sa_1 <- 1 - s_1_f

h13 <- lp_filter()
tc_2 <- h13 * sa_1

si_2 <- 1 - tc_2
s_2 <- M3X3_s * si_2
s_2_norm <- M2X12 * s_2
s_2_norm <- impute_last_obs(s_2_norm, n = 6, nperiod = 1)
s_2_demean <- s_2 - s_2_norm
s_2_f <- impute_last_obs(s_2_demean, n = 6, nperiod = 12)
sa_2 <- 1 - s_2_f
c(len = length(sa_2@sfilter), ub = upper_bound(sa_2@sfilter))

## len  ub
## 145  72
```

Exemple X-11 (2)

Contents

1. Introduction

2. rjd3filters et les moyennes mobiles

3. Autres packages

3.1 rjd3toolkit

3.2 Désaisonnalisation

3.3 Autres expérimentations

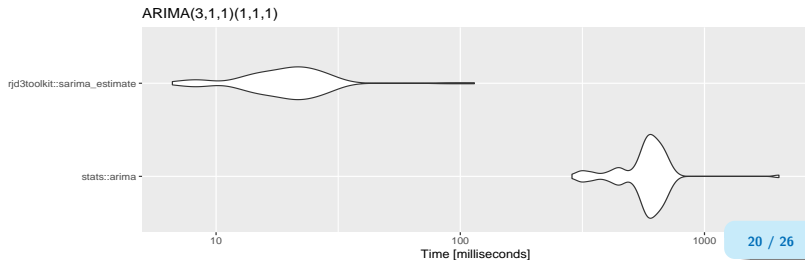
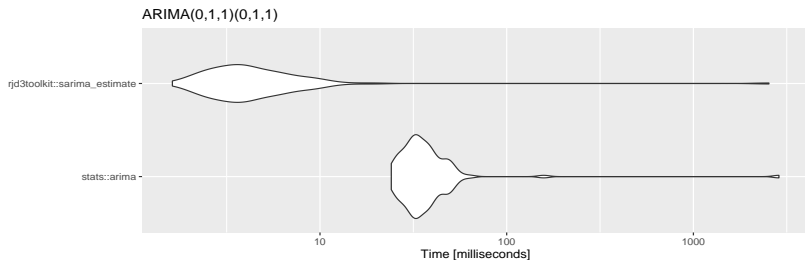
4. Conclusion

rjd3toolkit

- De nombreux tests (résidus, saisonnalité, effets jours ouvrables, etc.), non existants dans **R** : Jarque-Bera (plus de paramètres que rapport à `tseries::jarque.bera.test()`), *runs tests* (caractère aléatoire des données), *range-mean regression test* (choisir la transformation en log), tests sur les JO (Canova-Hansen), tests de saisonnalité (dont test combiné de X-11 et periodogramme)
- création de régresseurs liés aux points atypiques et liés au effets jours ouvrables (Pâques, année bissextile, stocks, variables trigonométriques, etc.)
- création de régresseurs de calendrier en prenant en compte les spécificités des pays (jours fériés)
- manipulation des modèles ARIMA (simulation, addition, décomposition, estimation)
- autres fonctions autour du traitement des séries temporelles (transformation, stationnarisation, etc.)

Benchmark des estimations ARIMA

En médiane 10 fois plus rapide avec $\text{ARIMA}(0,1,1)(0,1,1)$ et 30 fois avec $\text{ARIMA}(3,1,1)(1,1,1)$!

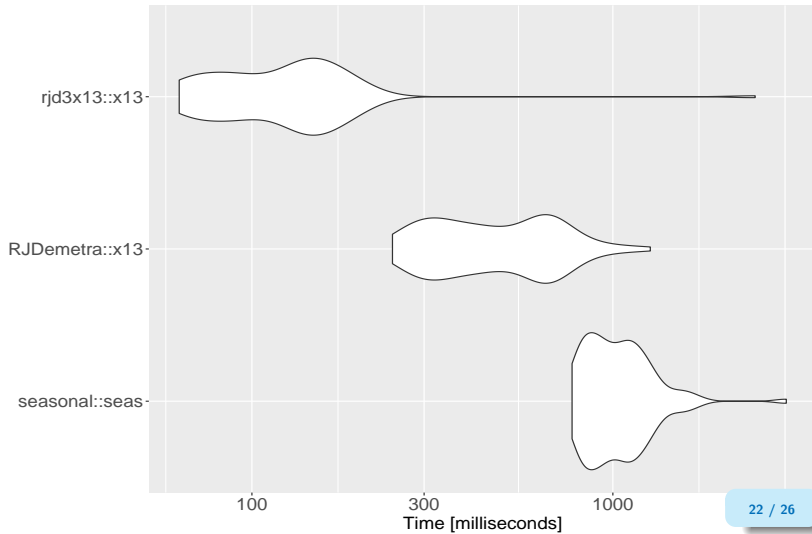


rjd3x13, rjd3tramoseats, rjdemetra3

- différentes méthodes de modélisation de RegARIMA avec detections automatiques des points atypiques
- méthodes “classiques de désaisonnalisation : X-13 et TRAMO-SEATS
- gestion de workspaces (liés à JDemetra+)



Performance

En médiane : RJDemetra 2 fois plus rapide que seasonal et rj3x13 10 fois plus rapide que seasonal !



rjd3highfreq et rjd3stl

rjd3highfreq Désaisonnalisation des séries haute-fréquence (plus fines que mensuelles)

-  fractional et multi airline decomposition
-  Extension de X-11 avec une saisonnalité non entière

rjd3stl : STL, MSTL, ISTL, loess

Exemples : https://github.com/palatej/test_rjd3hf

rjd3sts and rjd3bench

`rjd3sts` Interface simple autre des modèles espaces états et des *structural time series*

Exemples : https://github.com/palatej/test_rjd3sts

➡ `tvCoef` permet de transformer des modèles `lm` en modèles à coefficients variant dans le temps

`rjd3bench` Benchmarking (uni et multivarié) et temporal disaggregation

Exemples : https://github.com/palatej/test_rjd3bench

Contents


1. Introduction

2. rjd3filters et les moyennes mobiles

3. Autres packages


4. Conclusion

Conclusion

Avec JDemetra+ 3.0, de nombreux packages  sont développés :

- Sur l'analyse des séries temporelles et les méthodes de désaisonnalisation (bien plus rapide qu'avec les packages actuels)
- Développement de nouvelles méthodes (e.g. gestion des séries à haute-fréquence)
- Permet de faciliter les formations en accédant à toutes les fonctionnalités de JDemetra+
- Il faut $\mathfrak{L} \geq 17$ (non compatible avec tous les serveurs du CRAN)

Merci de votre attention

Packages  :

-  palatej/rjd3toolkit
-  palatej/rjd3x13
-  palatej/rjd3tramoseats
-  palatej/rjdemetra3
-  palatej/rjd3filters

-  palatej/rjd3sts
-  AQLT/tvCoef
-  palatej/rjd3stl
-  palatej/rjd3highfreq
-  palatej/rjd3bench