



Détection en temps réels des points de retournement : Apport de l'utilisation des filtres asymétriques dans l'analyse conjoncturelle

ALAIN QUARTIER-LA-TENTE

Maître de stage : OLIVIER DARNÉ (LEMNA)

26/10/2021

Ensaе — 2020-2021

Contexte

- Stage effectué au Laboratoire d'Économie et de Management de Nantes-Atlantique (LEMNA) avec Olivier Darné dans le cadre du début de ma thèse


Contexte

- Stage effectué au Laboratoire d'Économie et de Management de Nantes-Atlantique (LEMNA) avec Olivier Darné dans le cadre du début de ma thèse
- **Objectifs :**
 - Étudier et comparer les approches récentes pour l'extraction de la tendance-cycle en temps réel

Contexte

- Stage effectué au Laboratoire d'Économie et de Management de Nantes-Atlantique (LEMNA) avec Olivier Darné dans le cadre du début de ma thèse
- **Objectifs :**
 - Étudier et comparer les approches récentes pour l'extraction de la tendance-cycle en temps réel
 - Étudier les liens entre les méthodes avec une théorie générale (cf rapport)

Contexte

- Stage effectué au Laboratoire d'Économie et de Management de Nantes-Atlantique (LEMNA) avec Olivier Darné dans le cadre du début de ma thèse
- **Objectifs :**
 - Étudier et comparer les approches récentes pour l'extraction de la tendance-cycle en temps réel
 - Étudier les liens entre les méthodes avec une théorie générale (cf rapport)
 - Développer d'un package  (rjdfilters, <https://github.com/palatej/rjdfilters>, version en développement <https://github.com/AQLT/rjdfilters>)

Introduction

Une série X_t se décompose en plusieurs composantes inobservées :

$$X_t = \underbrace{TC_t}_{\text{tendance-cycle}} + \underbrace{S_t}_{\text{saisonnalité}} + \underbrace{I_t}_{\text{irrégulier}} \quad (\text{décomposition additive})$$

TC_t généralement estimée sur une série *sans* saisonnalité

Introduction

Une série X_t se décompose en plusieurs composantes inobservées :

$$X_t = \underbrace{TC_t}_{\text{tendance-cycle}} + \underbrace{S_t}_{\text{saisonnalité}} + \underbrace{I_t}_{\text{irrégulier}} \quad (\text{décomposition additive})$$

TC_t généralement estimée sur une série *sans* saisonnalité

Moyennes mobiles (ou *filtres linéaires*) omniprésents dans l'extraction de la tendance-cycle et la désaisonnalisation (e.g. : X-13ARIMA) :

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

Introduction

Une série X_t se décompose en plusieurs composantes inobservées :

$$X_t = \underbrace{TC_t}_{\text{tendance-cycle}} + \underbrace{S_t}_{\text{saisonnalité}} + \underbrace{I_t}_{\text{irrégulier}} \quad (\text{décomposition additive})$$

TC_t généralement estimée sur une série *sans* saisonnalité

Moyennes mobiles (ou *filtres linéaires*) omniprésents dans l'extraction de la tendance-cycle et la désaisonnalisation (e.g. : X-13ARIMA) :

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

➡ Généralement, utilisation de filtres *symétriques* ($p = f$ et $\theta_{-i} = \theta_i$)

Introduction

Une série X_t se décompose en plusieurs composantes inobservées :

$$X_t = \underbrace{TC_t}_{\text{tendance-cycle}} + \underbrace{S_t}_{\text{saisonnalité}} + \underbrace{I_t}_{\text{irrégulier}} \quad (\text{décomposition additive})$$

TC_t généralement estimée sur une série *sans* saisonnalité

Moyennes mobiles (ou *filtres linéaires*) omniprésents dans l'extraction de la tendance-cycle et la désaisonnalisation (e.g. : X-13ARIMA) :

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

➡ Généralement, utilisation de filtres *symétriques* ($p = f$ et $\theta_{-i} = \theta_i$)

➡ Pour l'estimation en **temps réel**, utilisation de filtres *asymétriques* ($f < p$) \implies révision et détection avec retard des points de retournement (*déphasage*) : cas du COVID-19

Introduction

Une série X_t se décompose en plusieurs composantes inobservées :

$$X_t = \underbrace{TC_t}_{\text{tendance-cycle}} + \underbrace{S_t}_{\text{saisonnalité}} + \underbrace{I_t}_{\text{irrégulier}} \quad (\text{décomposition additive})$$

TC_t généralement estimée sur une série *sans* saisonnalité

Moyennes mobiles (ou *filtres linéaires*) omniprésents dans l'extraction de la tendance-cycle et la désaisonnalisation (e.g. : X-13ARIMA) :

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

- ➡ Généralement, utilisation de filtres *symétriques* ($p = f$ et $\theta_{-i} = \theta_i$)
- ➡ Pour l'estimation en **temps réel**, utilisation de filtres *asymétriques* ($f < p$) \implies révision et détection avec retard des points de retournement (*déphasage*) : cas du COVID-19
- ➡ Comparaison de 3 méthodes qui pourraient être incluses dans X-13ARIMA

Sommaire

1. Introduction

2. Description des méthodes

2.1 Méthode actuelle

2.2 Polynômes Locaux

2.3 Filtres et Reproducing Kernel Hilbert Space (RKHS)

2.4 Minimisation sous contrainte : approche FST

3. Comparaison des méthodes

4. Conclusion

X-13ARIMA

1. Série étendue sur 1 an par un modèle ARIMA
2. Estimation de la tendance-cycle par moyenne mobile symétrique d'**Henderson**

X-13ARIMA

1. Série étendue sur 1 an par un modèle ARIMA
 2. Estimation de la tendance-cycle par moyenne mobile symétrique d'**Henderson**
- ➡ Prévisions combinaisons linéaires du passé : équivalent à utiliser des moyennes mobiles asymétriques avec coefficients optimisés pour minimiser les erreurs

X-13ARIMA

1. Série étendue sur 1 an par un modèle ARIMA
 2. Estimation de la tendance-cycle par moyenne mobile symétrique d'**Henderson**
- ➡ Prévisions combinaisons linéaires du passé : équivalent à utiliser des moyennes mobiles asymétriques avec coefficients optimisés pour minimiser les erreurs
- ➡ X-13ARIMA : décomposition itérative de X_T en TC_t , S_t et I_t avec une correction automatique des points atypiques

X-13ARIMA

1. Série étendue sur 1 an par un modèle ARIMA
 2. Estimation de la tendance-cycle par moyenne mobile symétrique d'**Henderson**
- ➔ Prévisions combinaisons linéaires du passé : équivalent à utiliser des moyennes mobiles asymétriques avec coefficients optimisés pour minimiser les erreurs
 - ➔ X-13ARIMA : décomposition itérative de X_T en TC_t , S_t et I_t avec une correction automatique des points atypiques
 - ➔ Comparaison de 3 approches modernes qui reproduise le filtre d'Henderson

Polynômes Locaux : `rjdfilters::lp_filter()`

Hypothèse : $y_t = \mu_t + \varepsilon_t$ avec $\varepsilon_t \stackrel{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$

μ_t localement approchée par un polynôme de degré d :

$$\forall j \in \llbracket -h, h \rrbracket : y_{t+j} = m_{t+j} + \varepsilon_{t+j}, \quad m_{t+j} = \sum_{i=0}^d \beta_{ij} j^i$$

Polynômes Locaux : `rjdfilters::lp_filter()`

Hypothèse : $y_t = \mu_t + \varepsilon_t$ avec $\varepsilon_t \stackrel{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$

μ_t localement approchée par un polynôme de degré d :

$$\forall j \in \llbracket -h, h \rrbracket : y_{t+j} = m_{t+j} + \varepsilon_{t+j}, \quad m_{t+j} = \sum_{i=0}^d \beta_i j^i$$

Estimation en utilisant les WLS avec *noyaux* : $\hat{\beta} = (X' K X)^{-1} X' K y$ et

$$\hat{m}_t = \hat{\beta}_0 = w' y = \sum_{j=-h}^h w_j y_{t-j} \Rightarrow \text{équivalent à une moyenne mobile symétrique}$$

➔ Filtre de Henderson avec $d = 3$ et noyau spécifique.

Filtres asymétriques : `rjdfilters::lp_filter()`

Plusieurs solutions :

1. Même méthode mais moins de données (DAF) \iff minimiser les révisions sous mêmes contraintes polynomiales
- ➡ **sans biais** mais **beaucoup de variance**

Filtres asymétriques : `rjdfilters::lp_filter()`

Plusieurs solutions :

1. Même méthode mais moins de données (DAF) \iff minimiser les révisions sous mêmes contraintes polynomiales

➡ **sans biais** mais **beaucoup de variance**

2. Minimisation des révisions sous contraintes polynomiales :

2.1 *Linear-Constant* (LC) : y_t linéaire and v reproduit les constantes (*Musgrave*)

2.2 *Quadratic-Linear* (QL) : y_t quadratique et v reproduit droites

2.3 *Cubic-Quadratic* (CQ) : y_t cubique et v reproduit tendances quadratiques

➡ Filtres asymétriques v dépendent de "IC-Ratio"

Filtres asymétriques : `rjdfilters::lp_filter()`

Plusieurs solutions :

1. Même méthode mais moins de données (DAF) \iff minimiser les révisions sous mêmes contraintes polynomiales

➡ **sans biais** mais **beaucoup de variance**

2. Minimisation des révisions sous contraintes polynomiales :

2.1 *Linear-Constant* (LC) : y_t linéaire and v reproduit les constantes (*Musgrave*)

2.2 *Quadratic-Linear* (QL) : y_t quadratique et v reproduit droites

2.3 *Cubic-Quadratic* (CQ) : y_t cubique et v reproduit tendances quadratiques

➡ Filtres asymétriques v dépendent de "IC-Ratio"



modèles simples facilement interprétables



Déphasage non contrôlé ➡ méthode étendue dans `rjdfilters::lp_filter()`

Filtres asymétriques : `rjdfilters::lp_filter()`

Plusieurs solutions :

1. Même méthode mais moins de données (DAF) \iff minimiser les révisions sous mêmes contraintes polynomiales

➡ **sans biais** mais **beaucoup de variance**

2. Minimisation des révisions sous contraintes polynomiales :

2.1 *Linear-Constant* (LC) : y_t linéaire and v reproduit les constantes (*Musgrave*)

2.2 *Quadratic-Linear* (QL) : y_t quadratique et v reproduit droites

2.3 *Cubic-Quadratic* (CQ) : y_t cubique et v reproduit tendances quadratiques

➡ Filtres asymétriques v dépendent de “IC-Ratio”



modèles simples facilement interprétables



Déphasage non contrôlé ➡ méthode étendue dans `rjdfilters::lp_filter()`

🖥 Visualisation <https://aqlt.shinyapps.io/FiltersProperties/>

Filtres RKHS : `rjdfilters::rkhs_filter()`

- Utilisation de la théorie des RKHS pour approcher le filtre d'Henderson
- Avec K_p une **fonction de noyau**, le filtre symétrique :

$$\forall j \in \llbracket -h, h \rrbracket : w_j = \frac{K_p(j/b)}{\sum_{i=-h}^h K_p(i/b)}$$

Filtres RKHS : `rjdfilters::rkhs_filter()`

- Utilisation de la théorie des RKHS pour approcher le filtre d'Henderson
- Avec K_p une **fonction de noyau**, le filtre symétrique :

$$\forall j \in \llbracket -h, h \rrbracket : w_j = \frac{K_p(j/b)}{\sum_{i=-h}^h K_p(i/b)}$$

- Pour les filtres asymétriques :

$$\forall j \in \llbracket -h, q \rrbracket : w_{a,j} = \frac{K_p(j/b)}{\sum_{i=-h}^q K_p(i/b)}$$

Filtres RKHS : `rjdfilters::rkhs_filter()`

- Utilisation de la théorie des RKHS pour approcher le filtre d'Henderson
- Avec K_p une **fonction de noyau**, le filtre symétrique :

$$\forall j \in \llbracket -h, h \rrbracket : w_j = \frac{K_p(j/b)}{\sum_{i=-h}^h K_p(i/b)}$$

➡ avec $b = h + 1$ et K_p spécifique on retrouve le filtre d'Henderson

- Pour les filtres asymétriques :

$$\forall j \in \llbracket -h, q \rrbracket : w_{a,j} = \frac{K_p(j/b)}{\sum_{i=-h}^q K_p(i/b)}$$

Filtres RKHS : `rjdfilters::rkhs_filter()`

- Utilisation de la théorie des RKHS pour approcher le filtre d'Henderson
- Avec K_p une **fonction de noyau**, le filtre symétrique :

$$\forall j \in \llbracket -h, h \rrbracket : w_j = \frac{K_p(j/b)}{\sum_{i=-h}^h K_p(i/b)}$$

➡ avec $b = h + 1$ et K_p spécifique on retrouve le filtre d'Henderson

- Pour les filtres asymétriques :

$$\forall j \in \llbracket -h, q \rrbracket : w_{a,j} = \frac{K_p(j/b)}{\sum_{i=-h}^q K_p(i/b)}$$

➡ b choisit par optimisation, e.g. minimisation du déphasage :

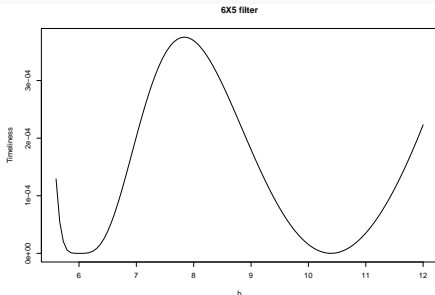
$$b_{q,\varphi} = \min_{b_q} \int_0^{2\pi/12} \rho_s(\lambda) \rho_\theta(\lambda) \sin^2 \left(\frac{\varphi_\theta(\omega)}{2} \right) d\omega$$

Filtres asymétriques



Plusieurs extremum

```
fun <- rkhs_optimization_fun(horizon = 6,
                             leads = 5, degree = 3,
                             asymmetricCriterion = "Timeliness")
plot(fun, 5.6, 12, xlab = "b",
     ylab = "Timeliness", main = "6X5 filter")
```



```
rkhs_optimal_bw()
```

```
##      q=0      q=1      q=2      q=3      q=4      q=5
## 6.0000 6.0000 6.3875 8.1500 9.3500 6.0000
```



Méthode
généralisable à des
filtres avec fréquences
irrégulières

Approche FST : `rjdfilters::fst_filter()`

Minimisation sous contrainte d'une somme pondérée de 3 critères :

$$\begin{cases} \min_{\theta} & J(\theta) = \alpha F_g(\theta) + \beta S_g(\theta) + \gamma T_g(\theta) \\ s.c. & C\theta = a \end{cases}$$

F_g fidélité (*fidelity*, réduction de variance), S_g lissage (*smoothness*, critère d'Henderson), T_g temporalité (*timeliness*, déphasage)

Approche FST : `rjdfilters::fst_filter()`

Minimisation sous contrainte d'une somme pondérée de 3 critères :

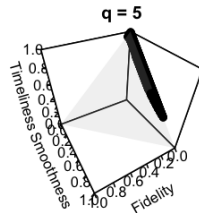
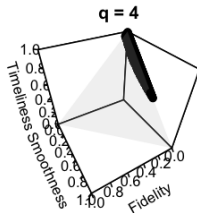
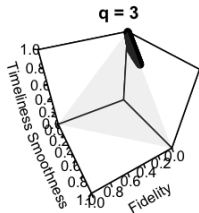
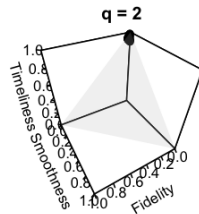
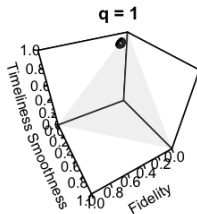
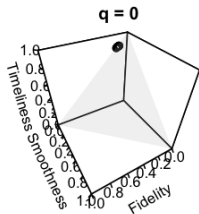
$$\begin{cases} \min_{\theta} & J(\theta) = \alpha F_g(\theta) + \beta S_g(\theta) + \gamma T_g(\theta) \\ \text{s.c.} & C\theta = a \end{cases}$$

F_g fidélité (*fidelity*, réduction de variance), S_g lissage (*smoothness*, critère d'Henderson), T_g temporalité (*timeliness*, déphasage)

- 😊 Solution unique
- 😊 Filtres asymétriques indépendants des données et du filtre symétrique
- ☹️ Poids non normalisés

Choix des poids

Idee : sélectionner les poids qui conduisent à des filtres qui minimise les 3 critères par rapport à une autre méthode (e.g., LC) sous les mêmes contraintes polynomiales



Sommaire

1. Introduction

2. Description des méthodes

3. Comparaison des méthodes

3.1 Méthodologie

3.2 Un exemple

3.3 Déphasage

4. Conclusion

Méthodologie

2 404 séries CJO (sts_inpr_m, IPI de l'UE) :

1. Désaisonnalisation avec X-13ARIMA (RJDemetra::x13) à chaque date pour extraire : série linéarisée, longueur des filtres saisonnier et tendance, schéma de décomposition et I-C ratio

Méthodologie

2 404 séries CJO (`sts_inpr_m`, IPI de l'UE) :

1. Désaisonnalisation avec X-13ARIMA (`RJDemetra::x13`) à chaque date pour extraire : série linéarisée, longueur des filtres saisonnier et tendance, schéma de décomposition et I-C ratio
2. Désaisonnalisation en **fixant** la série linéarisée et tous les autres paramètres et en utilisant un filtre spécifique pour la tendance-cycle (`rjdfilters::x11()`)

Méthodologie

2 404 séries CJO (`sts_inpr_m`, IPI de l'UE) :

1. Désaisonnalisation avec X-13ARIMA (`RJDemetra::x13`) à chaque date pour extraire : série linéarisée, longueur des filtres saisonnier et tendance, schéma de décomposition et I-C ratio
2. Désaisonnalisation en **fixant** la série linéarisée et tous les autres paramètres et en utilisant un filtre spécifique pour la tendance-cycle (`rjdfilters::x11()`)
3. À chaque date, estimation des points de retournement :
 - redressements : $y_{t-3} \geq y_{t-2} \geq y_{t-1} < y_t \leq y_{t+1}$
 - ralentissements : $y_{t-3} \leq y_{t-2} \leq y_{t-1} > y_t \geq y_{t+1}$

Méthodologie

2 404 séries CJO (`sts_inpr_m`, IPI de l'UE) :

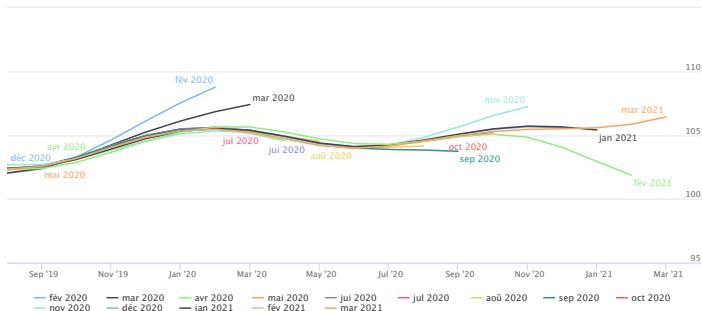
1. Désaisonnalisation avec X-13ARIMA (RJDemetra::x13) à chaque date pour extraire : série linéarisée, longueur des filtres saisonnier et tendance, schéma de décomposition et I-C ratio
2. Désaisonnalisation en **fixant** la série linéarisée et tous les autres paramètres et en utilisant un filtre spécifique pour la tendance-cycle (`rjdfilters::x11()`)
3. À chaque date, estimation des points de retournement :
 - redressements : $y_{t-3} \geq y_{t-2} \geq y_{t-1} < y_t \leq y_{t+1}$
 - ralentissements : $y_{t-3} \leq y_{t-2} \leq y_{t-1} > y_t \geq y_{t+1}$

Déphasage = temps nécessaire pour détecter le bon point de retournement sans révision

IPI fabrication de ciment, chaux et plâtre (C235) en Allemagne (point de retournement en février 2020)

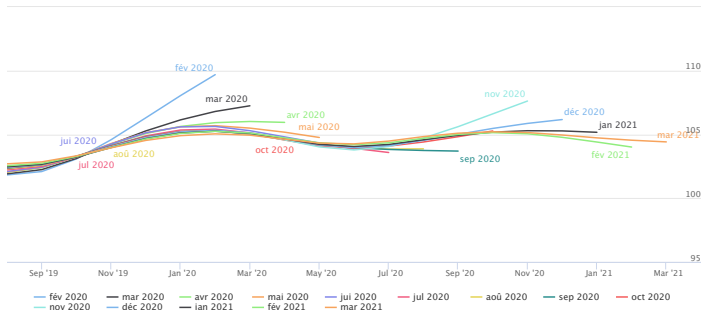
Tendance-cycle de la série C235_DE avec X-13-ARIMA

Déphasage = 13 mois



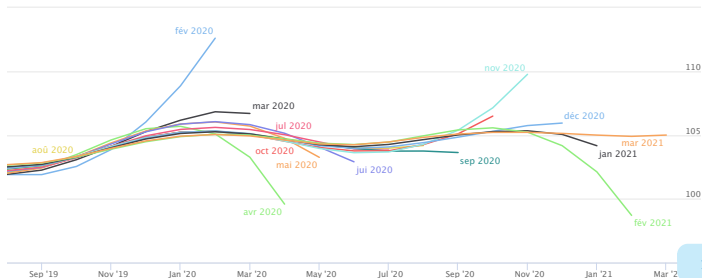
Tendance-cycle de la série C235_DE avec le filtre Linear-Constant (LC)

Déphasage = 3 mois



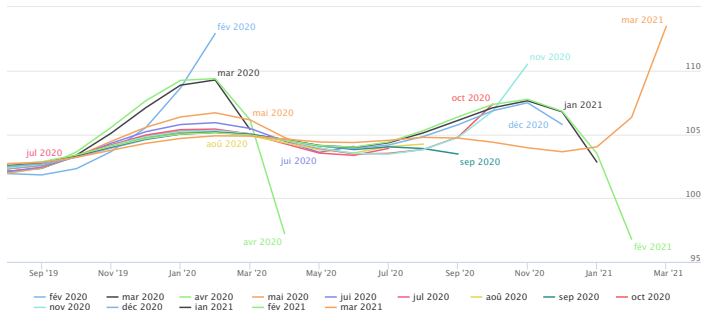
Tendance-cycle de la série C235_DE avec le filtre Quadratic-Linear (QL)

Déphasage = 3 mois



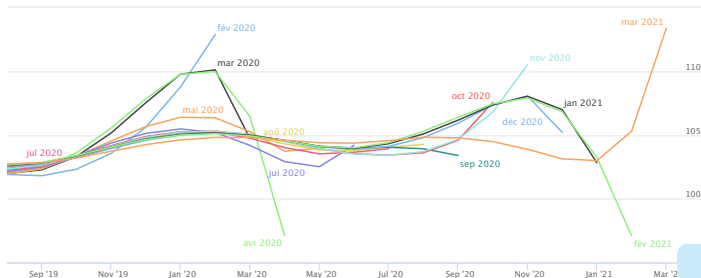
Tendance-cycle de la série C235_DE avec le filtre Cubic-Quadratic (CQ)

Déphasage = 2 mois



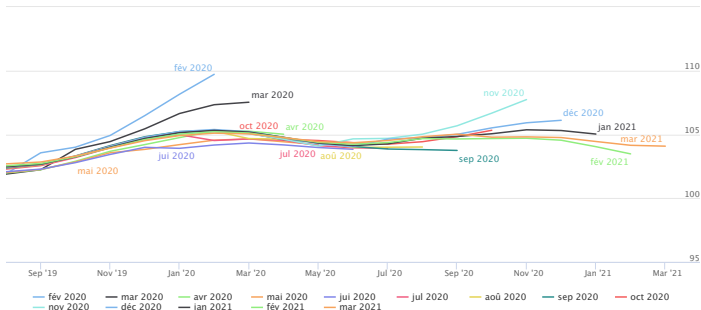
Tendance-cycle de la série C235_DE avec le filtre asymétrique direct (DAF)

Déphasage = 6 mois



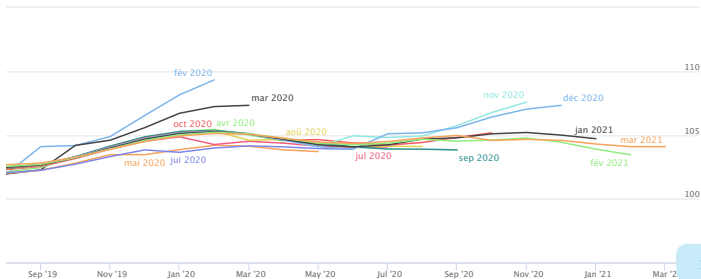
Tendence-cycle de la série C235_DE avec le filtre RKHS

Déphasage = 6 mois



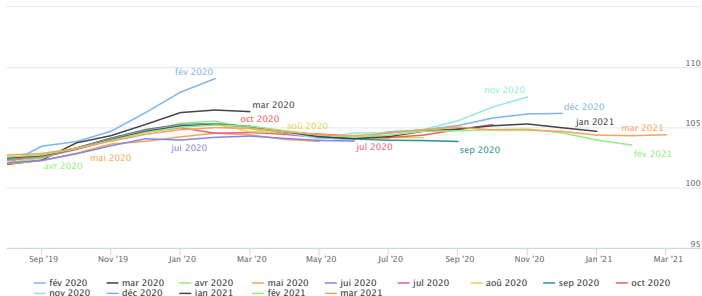
Tendence-cycle de la série C235_DE avec le filtre FST optimal par rapport au filtre LC et minimisant la timeliness

Déphasage = 7 mois



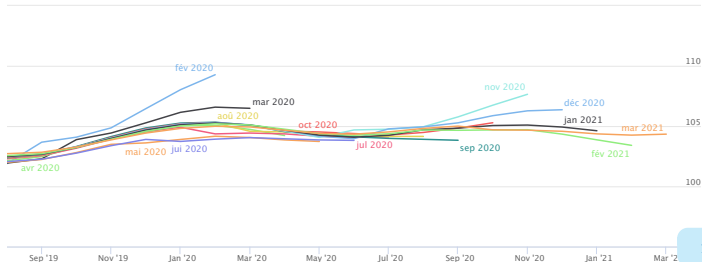
Tendance-cycle de la série C235_DE avec le filtre FST optimal par rapport au filtre LC et ayant la timeliness la plus élevée

Déphasage = 6 mois



Tendance-cycle de la série C235_DE avec le filtre FST optimal par rapport au filtre LC et ayant une timeliness médiane

Déphasage = 6 mois



Déphasage dans la détection de points de retournement en 2020

Pour les séries dont le filtre symétrique utilisé pour la tendance-cycle est de longueur 13 (771 séries)

	X-13ARIMA	LC	QL	CQ	DAF	$b_{q,\varphi}$	FST - LC		
							Min.	Max.	Méd.
Min	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0
Q1	3,0	4,0	3,0	2,0	2,0	6,0	6,0	6,0	6,0
Median	4,0	4,0	4,0	6,0	6,0	6,0	7,0	6,0	6,0
Q3	5,0	5,0	7,0	7,0	7,0	7,0	7,0	6,0	7,0
Max	14,0	14,0	14,0	14,0	14,0	14,0	14,0	14,0	14,0
Mean	4,5	4,7	5,1	5,4	5,5	6,6	7,0	6,5	6,6

Déphasage dans la détection de points de retournement en 2020

Pour les séries dont le filtre symétrique utilisé pour la tendance-cycle est de longueur 13 (771 séries)

	X-13ARIMA	LC	QL	CQ	DAF	$b_{q,\varphi}$	FST - LC		
							Min.	Max.	Méd.
Min	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0
Q1	3,0	4,0	3,0	2,0	2,0	6,0	6,0	6,0	6,0
Median	4,0	4,0	4,0	6,0	6,0	6,0	7,0	6,0	6,0
Q3	5,0	5,0	7,0	7,0	7,0	7,0	7,0	6,0	7,0
Max	14,0	14,0	14,0	14,0	14,0	14,0	14,0	14,0	14,0
Mean	4,5	4,7	5,1	5,4	5,5	6,6	7,0	6,5	6,6

Déphasage dans la détection de points de retournement en 2020

Pour les séries dont le filtre symétrique utilisé pour la tendance-cycle est de longueur 13 (771 séries)

							FST - LC		
	X-13ARIMA	LC	QL	CQ	DAF	$b_{q,\varphi}$	Min.	Max.	Méd.
Min	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0
Q1	3,0	4,0	3,0	2,0	2,0	6,0	6,0	6,0	6,0
Median	4,0	4,0	4,0	6,0	6,0	6,0	7,0	6,0	6,0
Q3	5,0	5,0	7,0	7,0	7,0	7,0	7,0	6,0	7,0
Max	14,0	14,0	14,0	14,0	14,0	14,0	14,0	14,0	14,0
Mean	4,5	4,7	5,1	5,4	5,5	6,6	7,0	6,5	6,6

Déphasage dans la détection de points de retournement en 2020

Pour les séries dont le filtre symétrique utilisé pour la tendance-cycle est de longueur 13 (771 séries)

	X-13ARIMA	LC	QL	CQ	DAF	FST - LC			
						$b_{q,\varphi}$	Min.	Max.	Méd.
Min	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0
Q1	3,0	4,0	3,0	2,0	2,0	6,0	6,0	6,0	6,0
Median	4,0	4,0	4,0	6,0	6,0	6,0	7,0	6,0	6,0
Q3	5,0	5,0	7,0	7,0	7,0	7,0	7,0	6,0	7,0
Max	14,0	14,0	14,0	14,0	14,0	14,0	14,0	14,0	14,0
Mean	4,5	4,7	5,1	5,4	5,5	6,6	7,0	6,5	6,6

Conclusion

- Dans la construction des filtres asymétriques, on peut se restreindre à ceux qui conserve les polynômes de degré au plus 1 (et exclure les filtres QL, CQ et DAF)

Conclusion

- Dans la construction des filtres asymétriques, on peut se restreindre à ceux qui conserve les polynômes de degré au plus 1 (et exclure les filtres QL, CQ et DAF)
- Durant la crise du COVID-19, la méthode actuelle X-13ARIMA semble satisfaisante en moyenne. . .

Conclusion

- Dans la construction des filtres asymétriques, on peut se restreindre à ceux qui conserve les polynômes de degré au plus 1 (et exclure les filtres QL, CQ et DAF)
- Durant la crise du COVID-19, la méthode actuelle X-13ARIMA semble satisfaisante en moyenne. . . mais peut provenir des définitions, indicateurs et méthodologie utilisés

Conclusion

- Dans la construction des filtres asymétriques, on peut se restreindre à ceux qui conserve les polynômes de degré au plus 1 (et exclure les filtres QL, CQ et DAF)
- Durant la crise du COVID-19, la méthode actuelle X-13ARIMA semble satisfaisante en moyenne. . . mais peut provenir des définitions, indicateurs et méthodologie utilisés
- Dans certains cas des filtres alternatifs peuvent aider ➡ `rjdfilters` peut aider à comparer les résultats

What next?

- Comprendre quand et pourquoi une méthode est plus performante qu'une autre

What next?

- Comprendre quand et pourquoi une méthode est plus performante qu'une autre
- Etudes sur d'autres données avec d'autres méthodes


What next?

- Comprendre quand et pourquoi une méthode est plus performante qu'une autre
- Etudes sur d'autres données avec d'autres méthodes
- Utiliser des paramètres différents en fin de période?

What next?

- Comprendre quand et pourquoi une méthode est plus performante qu'une autre
- Etudes sur d'autres données avec d'autres méthodes
- Utiliser des paramètres différents en fin de période?
- Impact des points atypiques? quid des méthodes robustes?

Merci pour votre attention

Package  :

 palatej/rjdfilters

 Rapport en ligne : <https://aqlt-stage3a.netlify.app/>