Installation des outils pour la désaisonnalisation

Tanguy BARTHELEMY

2023-01-17

Contexte

Les outils d'aide à la désaisonnalisation sont :

- R et Rstudio
- JDemetra+

⚠ Aucune assistance en langage **SAS** n'est apportée. ⚠

Préalable

Sur les ordinateurs sans droit d'administrateur (*postes professionnels par exemple*), il est conseillé de créer un dossier **Software** sous C:\Users\...\Software ou directement sous C:\Users\Software dans lequel tous les logiciels seront installés.

Attention : lorsqu'on précise un chemin **absolu** de logiciel (JDemetra+, Java, R, ...) dans un programme, un raccourci, une variable, ..., il doit être modifié à chaque fois qu'un des répertoires racines du logiciel est déplacé.

1 Installation de JDemetra+

1.1 Version de JDemetra+ et dépendance

JDemetra+ est téléchargeable depuis le lien github de l'application : https://github.com/jdemetra/jdemetra-app/releases.

La dernière release (v2.2.3) date du 7 juillet 2020. C'est la dernière version **stable** de JDemetra+. **C'est cette version qu'il convient de télécharger et d'utiliser en production.**

Il existe 2 autres versions de JDemetra+ qui en sont uniquement au stade de **test** :

- v2.2.4 : une v2.2.3 corrigée des bugs mais sans ajout de nouvelles fonctionnalités majeures
- v3.0.0 : la nouvelle version de JDemetra+ avec de nouvelles fonctionnalités et une nouvelle interface

Les v2.2.3 et v2.2.4 de JDemetra+ nécéssitent une version de Java >= 8 alors que la v3.0.0 nécessite une version de Java >= 17 :

JDemetra+ version	Java version
v2.2.3	>= 8
v2.2.4 (RC2)	>= 8
v3.0.0 (RC1)	>= 17

RC : Release candidate

Ainsi les processus d'installation de ces 3 versions sont semblables. Seulement v2.2.3 et v2.2.4 nécessite une version de Java >= 8 alors que v3.0.0 nécessite une version de Java >= 17.

1.2 Processus d'installation

Deux solutions pour l'installer :

- **Télécharger et exécuter** le fichier `.exe` qui nécessite des droits d'administrateur
- **Télécharger** le `.zip` qui permet d'avoir une version portable du logiciel.

Attention : pour la seconde option il faut télécharger le fichier `jdemetra+-2.2.3-bin.zip` (pour la version 2.2.3 par exemple) et le fichier `Source code (zip)` :

Une fois le fichier téléchargé, il suffit de le dézipper : le logiciel se trouve alors dans le dossier \nbdemetra\bin\, ce sont les fichiers `nbdemetra.exe` (version 32-bit) et nbdemetra64.exe (version 64-bit).

Conseil : si vous comptez utiliser plusieurs version de JDemetra+ (v2.2.3, v3.0.0, ...), vous pouvez renommer les dossiers dézippé en \nbdemetra-2.2.3\, \nbdemetra-2.2.4\ et \nbdemetra-3.0.0\.

Remarque : Vous pouvez créer des raccourcis des exécutables si vous souhaitez les lancer depuis le bureau ou d'autres dossiers.

1.3 Installation du cruncher

Le cruncher (**JWSACruncher**) est un outil qui permet de mettre à jour un workspace de JDemetra+ à partir de la console, sans devoir ouvrir JDemetra+. La mise à jour d'un workspace peut alors se faire à partir d'un autre logiciel statistique (R ou SAS par exemple). La version 2.2.3 du cruncher peut être téléchargée ici https://github.com/jdemetra/jwsacruncher/releases (fichier `jwsacruncher-x.y.z-bin.zip`) et les anciennes versions sont téléchargeables sous https://github.com/jdemetra/jdemetra-core/releases.

Si l'on utilise une version portable de Java (voir section installation Java), il faut encore modifier certains paramètres pour utiliser le cruncher :

- **Dézipper** le fichier téléchargé
- dans le dossier dézippé, **ouvrir** (par exemple avec Notepad++) le fichier `jwsacruncher.bat` présent dans le sous-dossier \bin\ (c'est-à-dire sous jdemetra-cli-2.2.3\bin\ dans la version 2.2.3 du cruncher)
- **Modifier** la valeur de la variable `JAVACMD` de la ligne **71** (actuelle `JAVACMD=java`) par l'adresse vers le fichier `java.exe` de la version portable . Ainsi, si JPortable est installé sous C:\Users\Software, la nouvelle ligne est `if "%JAVACMD%"=="" set JAVACMD="C:\\Users\\Software\\Java64\\bin\\java"` (pour Java 8).

2 Installation de Java

2.1 Java 8

Pour installer Java 8, utiliser le lien https://portableapps.com/apps/utilities/java_portable.

2.2 Java 17

Pour installer Java 17, il faut aller à l'adresse https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html.

- **Télécharger** la version **Compressed Archive** de Windows (https://download.oracle.com/java/17/archive/jdk-17.0.4.1_windows-x64_bin.zip)
- **Dézipper** le dossier jdk-17.0.6 sous C:\Users\Software (par exemple)

Après avoir installé Java (en version 8, 17 ou autre), il faut :

- **modifier** les variables d'environnement de PATH de Rstudio et de Windows (voir section Variables d'environnement)
- **modifier** les cibles de JDemetra+ pour informer de la localisation des nouvelles versions de Java.

Ainsi par exemple, si vous avez installé la version 17 de Java pour utiliser la version 3.0.0 de JDemetra+. Il est nécessaire d'ajouter la localisation de Java 17 au raccourci de l'exécutable grace à l'option `--jdkhome`. La cible du raccourci devient `C:\Users\Software\nbdemetra-3.0.0\bin\nbdemetra64.exe --jdkhome "C:\Users\Software\Java17\jdk17"`

3 Installation de R et Rstudio

Les fonctionnalités de JDemetra+ sont accessibles sur R via des packages R. Pour utiliser R, mieux vaut utiliser un IDE donc Rstudio. Tous les exécutables à télécharger sont sous https://posit.co/download/rstudio-desktop/#download.

3.1 Installation de R

Pour installer R, il faut :

- **Télécharger** le fichier binaire `R-4.2.2-win.exe` sous https://cran.rstudio.com/bin/windows/base/
- **Exécuter** l'exécutable pour paramétrer et **installer** R.

3.2 Installation de Rstudio

Télécharger la dernière version de Rstudio (sous https://posit.co/download/rstudio-desktop/#download) et l'**installer**.

3.3 Installation des packages R

3.3.1 En version 2

Les packages en version 2 sont les suivants :

Nom	Disponible sur le CRAN	Lien Github
RJDemetra	✅	https://github.com/jdemetra/rjdemetra
rjdworkspace	❌	https://github.com/InseeFrLab/rjdworkspace
JDCruncheR	❌	https://github.com/InseeFr/JDCruncheR

Le code d'installation des packages est ci-dessous :

```
# If remotes is not installed
# install.packages("remotes")

remotes::install.packages("RJDemetra")
remotes::install_github("InseeFrLab/rjdworkspace")
remotes::install_github("InseeFr/JDCruncheR")
```

3.3.2 En version 3

Pour l'instant aucun package de la version 3 n'est sur le CRAN. Pour installer les packages :

```
# If remotes is not installed
# install.packages("remotes")
```

```r
remotes::install_github("palatej/rjdemetra3")

remotes::install_github("palatej/rjd3toolkit")
remotes::install_github("palatej/rjd3modelling")
remotes::install_github("palatej/rjd3sa")
remotes::install_github("palatej/rjd3arima")
remotes::install_github("palatej/rjd3x13")
remotes::install_github("palatej/rjd3tramoseats")
remotes::install_github("palatej/rjdemetra3")
remotes::install_github("palatej/rjdfilters")
remotes::install_github("palatej/rjd3sts")
remotes::install_github("palatej/rjd3highfreq")
remotes::install_github("palatej/rjd3stl")
remotes::install_github("palatej/rjd3bench")

remotes::install_github("AQLT/ggdemetra3")
```

3.3.3 Cas AUS

Pour installer un package sur AUS,

- soit il se trouve sur le CRAN et il est possible de l'installer directement via la fonction `install.packages()`
- soit il ne s'y trouve pas et il faut alors récupérer le package au format binaire (.zip) et l'installer avec la fonction `install.packages()` et les arguments `repos = NULL, type = "binary"`.

Exemple pour le package **JDCrunchR** :

```r
install.packages("chemin/.../JDCruncheR_0.2.4.tar.gz",
                 repos = NULL, type = "binary")
```

4 Variables d'environnement

4.1 Sous Rstudio

4.1.1 Proxy

Pour les postes Insee, il faut **configurer** le proxy et des paramètres de localisation des logiciels sous Rstudio. Il suffit de :

- **Lancer** le code `file.edit("~/.Renviron")`
- **Ajouter** les paramètres (nouvelles lignes) :

  ```
  http_proxy = http://proxy-rie.http.insee.fr:8080/
  https_proxy = http://proxy-rie.http.insee.fr:8080/
  ```

- **Sauvegarder** et **fermer** le fichier

4.1.2 Version de Java

Si une nouvelle version de Java a été installé, il faut préciser à Rstudio la localisation de l'installation. Pour cela, comme pour paramétrer le proxy :

- **Lancer** le code `file.edit("~/.Renviron")`
- **Ajouter** les paramètres :

  ```
  JAVA_HOME = "C:/Users/Software/Java17/jdk17"
  ```

- **Sauvegarder** et **fermer** le fichier

4.1.3 PATH

La variable d'environnement `PATH` en R sert à indiquer à R où chercher les fichiers exécutables.

Pour modifier cette variable d'environnement :

- **Récupérer** les chemins de logiciels déjà enregistrés via la commande `Sys.getenv("PATH")` (Rstudio renvoie alors une succession d'adresses du type `C:/WINDOWS/system32;C:/WINDOWS`)
- **Copier-coller** ces adresses après `PATH =` et y ajouter les chemins vers les répertoires \bin\ (binary) des logiciels nouvellement installés, en les séparant par des points-virgules sans espace avant ni après. Par exemple, pour l'installation de Rtools, le chemin est C:\rtools42\mingw64\bin (selon là où a été installé Rtools). Il faut donc rajouter `C:\\rtools42\\mingw64\\bin` ou `C:/rtools42/mingw64/bin` (En `R`, `\` est un caractère spécial, donc il faut remplacer les `\` par `/` ou par `\\`). Le chemin devient `C:/WINDOWS/system32;C:/WINDOWS;C:/rtools42/mingw64/bin`.
- **Modifier** la variable avec la fonction `Sys.setenv()`. Pour l'exemple ci-dessus, la commande à lancer est :

```
Sys.setenv(PATH = "C:/WINDOWS/system32;C:/WINDOWS;C:/rtools42/mingw64/bin")
```

NB : Généralement une version 32 bits et une version 64 bits sont disponible au téléchargement et à l'installation d'un logiciel. Il faut choisir les dossiers propres à sa version système :

Version	Mots- clés
64 bits	64 x64
32 bits	32 x86

Plus d'infos sur la variable PATH via la page https://java.com/fr/download/help/path.xml.

5 Vérifications

Pour s'assurer que tout fonctionne bien, on peut faire tourner des exemples de code et vérifier qu'il n'y a pas d'erreurs :

```
library("RJDemetra")

myseries <- ipi_c_eu[, "FR"]
x13_model <- x13(myseries) # X-13ARIMA method
ts_model <- tramoseats(myseries) # TRAMO-SEATS method

# Basic plot with the original series, the trend and the SA series
plot(x13_model, type_chart = "sa-trend")
```

Pour vérifier la version de Java que l'on utilise sous R, on peut essayer d'installer et utiliser le package `rJava` et lancer la commande ci-dessous :

```
# If rJava is not installed
install.packages("rJava")
```

Si l'installation de `rJava` retourne une erreur, cela veut dire que Java a été mal installé ou mal configuré sur R. Il faut retourner à la section Variables d'environnement.

Ce bloc de commande test la version de Java avec laquelle R fonctionne :

```
library("rJava")
.jinit()
.jcall("java/lang/System", "S", "getProperty", "java.runtime.version")
```

Enfin lon peut consulter la version de Java installé sur notre poste et avec laquelle windows fonctionne (cela n'a pas d'importance pour nous) :

```
system("java -version")
```

6 Installations optionnelles

Certaines installation supplémentaires sotn optionnelles (c'est-à-dire qu'elles ne sont pas obligatoires mais apportent des fonctionnalités externes) :

- Miktek pour produire des documents latex
- Rtools pour développer des packages et compiler du code

7 Problèmes que l'on peut rencontrer

7.1 Problème d'installation de package R

Si lors de l'installation de packages, vous obtenez l'erreur suivante :

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

```
## Error in eval(expr, envir, enclos): Erreur : le chargement a échoué
## Exécution arrêtée
## *** arch - x64
```

Le problème ne vient pas de Java mais du package R. Par défaut, le package s'installe depuis le fichier "source", c'est-à-dire que le package est recompilé. Pour certaines raisons informatiques, lorsqu'on compile par défaut, ce sont les paramètres du système qui sont utilisés (et qui n'ont pas forcément les bonnes versions de Java).

Deux solutions :

- ne pas compiler le package en installant depuis le fichier "binary" :

  ```
  install.packages("RJDemetra", type = "binary")
  ```

- spécifier que l'on veut utiliser les paramètres locaux :

  ```
  install.packages("RJDemetra", type = "source", INSTALL_opts = "--no-multiarch")
  ```

Plus d'information : https://github.com/jdemetra/rjdemetra/wiki/Installation-manual

7.2 La commande `library("RJDemetra")` renvoie un message d'erreur

Le package **RJDemetra** a besoin de la version 8 (au minimum) de Java pour fonctionner, qui n'est pas celle utilisée par défaut sous AUS (au 3 mars 2020). Si au moins un autre package a déjà été chargé via la fonction `library()` et qu'il ne nécessite pas une version très à jour de Java, c'est cette ancienne version qui sera sollicitée pendant toute la durée de la session (R est réfractaire au changement de version en cours de session). En cas d'utilisation de **RJDemetra** au cours d'un programme, il faut donc impérativement spécifier dès le début du programme qu'il faut que R ailler chercher la version 8, via la commande :

```
Sys.setenv(JAVA_HOME = "C:/Users/Software/Java17/jdk17") #Là où est installé java
```

ou charger RJDemetra en premier

```
# En début de programme
library("RJDemetra")
```

Sinon il faut redémarrer une nouvelle session R.

7.3 `Error array index = -1`

Le message du type `Error array index = -1` indique une variable auxiliaire non trouvée. Il peut s'agir de régresseurs CJO ou d'autres variables définies par l'utilisateur (effet de Pâques spécifique, PSO = pure seasonal outlier…).