Configurer JDemetra+ et les outils R associés

Tanguy BARTHELEMY

Contexte

Ce document complète le [guide d'installation rapide] (https://github.com/TanguyBarthelemy/JD-Tutorials/blob/master/Fiches/Installation_rapide_JDemetra+.pdf) en fournissant plus de détails sur la configuration requise pour faire fonctionner JDemetra+ et les outils associés.

Sur les ordinateurs sans droits d'administrateur (*ordinateur professionnel, par exemple*), il est recommandé de créer un dossier **Software** sous C:\Users\...\Software ou directement sous C:\Users\Software où tous les logiciels seront installés.

⚠ Attention : lorsque nous spécifions un chemin **absolu** pour un logiciel (JDemetra+, Java, **R**, …) dans un programme, un raccourci, une variable, …, il doit être modifié à chaque fois qu'un dépôt racine est déplacé.

0.1 Installation de JDemetra+ Interface graphique

Pour installer JDemetra+ Graphique User Interface voir le [guide d'installation rapide] (https://github.com/TanguyBarthelemy/JD-Tutorials/blob/master/Fiches/Installation_rapide_JDemetra+.pdf).

0.2 Installation du cruncher

Pour installer JDemetra+ cruncher voir le [guide d'installation rapide] (https://github.com/TanguyBarthelemy/JD-Tutorials/blob/master/Fiches/Installation_rapide_JDemetra+.pdf)

0.3 Installation de R et Rstudio

Les algorithmes de JDemetra+ sont accessibles dans **R** via les packages R.

Pour utiliser **R**, il est préférable d'utiliser un IDE comme Rstudio. Tous les fichiers exécutables à télécharger se trouvent sous https://posit.co/download/rstudio-desktop/#download.

0.3.1 Installation de R

Pour installer **R**, vous devez

- **Télécharger** le fichier binaire `R-4.3.2-win.exe` sous https://cran.rstudio.com/bin/windows/base/
- **Exécuter** l'exécutable pour paramétrer et installer **R**.

0.3.2 Installation de Rstudio

Pour installer Rstudio, il suffit de **téléchargez** la dernière version de Rstudio (sous https://posit.co/download/rstudio-desktop/#download) et le **installateur**.

Si l'installation via le fichier `.exe` échoue (parce qu'elle nécessite des droits plus élevés (administrateur, élévation, ...), nous obtiendrons une version portable du logiciel. Pour ce faire :

- **Télécharger** et **décompresser** le dossier compressé `.zip` dans un dossier nommé "Rstudio" (sous C:\Users\Software)
- **Créer un raccourci** du fichier `rstudio.exe` sur le Bureau.

0.3.3 Installation des packages R

Pour installer un package **R**, il y a plusieurs méthodes :

- Soit il est sur CRAN et vous pouvez l'installer directement avec la fonction `install.packages()`.
- Soit il est sur Github et vous pouvez l'installer directement avec la fonction `install_github()` à partir du package **remotes**.
- Soit vous devez récupérer le package dans un dossier (format binaire) (`.zip`) et ensuite l'installer avec la fonction `install.packages()` avec les arguments `repos = NULL, type = "binary"`.

0.3.3.1 Les packages R pour JDemetra+ version 2.2.4

Les packages de la version 2 sont :

Nom	Sur le CRAN	Sur AUS	Lien Github
RJDemetra	✅	✅	https://github.com/jdemetra/rjdemetra
rjdworkspace	❌	✅	https://github.com/InseeFrLab/rjdworkspace
JDCruncheR	❌	✅	https://github.com/InseeFr/JDCruncheR
rjwsacruncher	✅	✅	https://github.com/AQLT/rjwsacruncher
rjdmarkdown	✅	✅	https://github.com/AQLT/rjdmarkdown

Le code d'installation des packages se trouve ci-dessous :

```
# Si le package remotes n'est pas installé
# install.packages("remotes")

install.packages("RJDemetra")
install.packages("rjwsacruncher")
install.packages("rjdmarkdown")

remotes::install_github("InseeFrLab/rjdworkspace")
remotes::install_github("InseeFr/JDCruncheR")

# Sous AUS et sur les ordinateurs Insee
install.packages("rjdworkspace", repos = "https://nexus.insee.fr/repository/r-public")
install.packages("JDCruncheR", repos = "https://nexus.insee.fr/repository/r-public")
```

0.3.3.2 Packages R pour JDemetra+ version 3.x

Actuellement, les packages pour la version 3 ne sont PAS sur CRAN. Pour les installer, vous devez passer par Github :

```
# Si le package remotes n'est pas installé
# install.packages("remotes")

remotes::install_github("rjdemetra/rjd3toolkit")

remotes::install_github("rjdemetra/rjd3x13")
remotes::install_github("rjdemetra/rjd3tramoseats")

remotes::install_github("rjdemetra/rjd3providers")
remotes::install_github("rjdemetra/rjdemetra3")

remotes::install_github("rjdemetra/rjd3filters")
remotes::install_github("rjdemetra/rjd3x11plus")

remotes::install_github("rjdemetra/rjd3sts")
remotes::install_github("rjdemetra/rjd3highfreq")
remotes::install_github("rjdemetra/rjd3stl")

remotes::install_github("rjdemetra/rjd3revisions")
remotes::install_github("rjdemetra/rjd3bench")
remotes::install_github("rjdemetra/rjd3nowcasting")

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

0.3.3.3 Cas AUS (Insee)

Pour installer un package sur AUS, il n'est pas possible d'utiliser la fonction `install_github()`. Ainsi :

- soit le package est sur le nexus d'AUS, il peut être installé avec la fonction `install.packages()` et l'argument `repos = "https://nexus.insee.fr/repository/r-public/"`
- soit il n'est pas disponible et doit être téléchargé au format binaire. Pour cela, il faut aller chercher le dossier compressé `.zip` sous GitHub.

Exemple pour le package **{rjd3toolkit}**, on peut installer le package :

- sur le nexus avec le code suivant :

```
install.packages("rjd3toolkit", repos = "https://nexus.insee.fr/repository/r-public/")
```

- au format binaire `rjd3toolkit_3.2.2.zip` qui se trouve sous https://github.com/rjdemetra/rjd3toolkit/releases/tag/v3.2.2 (*release* Section). Après l'avoir récupéré, il faut lancer la commande d'installation :

```
install.packages("chemin/vers/le/binaire/.../rjd3toolkit_3.2.2.zip ",
                 repos = NULL, type = "binary")
```

0.4 Variables d'environnement

0.4.1 Dans Rstudio

Pour ajouter des variables d'environnement dans Rstudio, vous devez ajouter le nom et la valeur de la variable. Il y a deux façons de le faire :

- En utilisant le fichier `.Renviron` :
- **Lancer** le code `file.edit("~/.Renviron")` ou `usethis::edit_r_environ()` (avec le package **usethis** et l'argument `scope` qui égalise `"user"` ou `"project"` si vous êtes dans un projet R).
- Ajoutez les variables au fichier (avec de nouvelles lignes).
- Sauvegardez le fichier
- En utilisant le fichier `.Rprofile` :
- **Lancer** le code `file.edit("~/.Rprofile")` ou `usethis::edit_r_profile()` (avec le package **usethis** et l'argument `scope` qui égalise `"user"` ou `"project"` si vous êtes dans un projet R)
- Ajoutez les variables avec la fonction `Sys.setenv()` (avec de nouvelles lignes)
- Sauvegardez le fichier

0.4.2 PATH

La variable d'environnement `PATH` dans **R** est utilisée pour indiquer à **R** où trouver les fichiers exécutables.

Lorsque vous installez un nouveau logiciel (par exemple JDemetra+, Rtools, Java…) que Rstudio utilise, vous devez modifier cette variable d'environnement :

- **Obtenez** la valeur actuelle de la variable `PATH` via la commande **R** `Sys.getenv("PATH")` (Rstudio renvoie une succession d'adresses comme `C:/WINDOWS/system32;C:/WINDOWS`)
- **Copier coller** cette valeur après `PATH =` et ajouter les chemins vers le dossier \bin\ (binaire) du logiciel nouvellement installé, en les séparant par un point-virgule (sans espace avant ou après).

Pour l'installation de Rtools le chemin estC:\rtools42\mingw64\bin (cela dépend de l'endroit où Rtools a été installé). Il vous faut ajouter `C:\\rtools42\\mingw64\\bin` or `C:/rtools42/mingw64/bin` (Dans **R**, `\` est un caractère spécial, il faut remplacer `\` par `/` ou par`\\`). Le chemin devient `C:/WINDOWS/system32;C:/WINDOWS;C:/rtools42/mingw64/bin`.

- **Modifiez** la variable avec la fonction `Sys.setenv()`. Pour Rtools, la commande à lancer est la suivante :

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

ℹ NB : En général, une version 32 bits et une version 64 bits sont disponibles pour le téléchargement et l'installation d'un logiciel. Vous devez vérifier le type de processeur de votre système d'exploitation pour choisir le bon dossier à télécharger.

Pour cela, vous pouvez lancer la commande :

```
Sys.getenv("R_ARCH")
Sys.info()[["machine"]]
```

D'après le résultat, la version est 32 bits ou 64 bits :

Version	Output
64 bits	/x64 x86-64
32 bits	/i386 x86_32

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

0.5 Vérifiez votre configuration

Pour vous assurer que tout fonctionne correctement, vous pouvez lancer un exemple de code **R** et vérifier qu'il n'y a pas d'erreur :

```
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 nous utilisons sur **R**, vous pouvez essayer d'installer et d'utiliser le package **rJava** et lancer la commande ci-dessous :

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

Si l'installation de **rJava** renvoie une erreur, cela signifie que Java a été mal installé ou mal configuré sur **R**. Vous devez revenir à la section Variables d'environnement.

Ce bloc de code teste la version de Java avec laquelle **R** fonctionne :

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

Enfin, vous pouvez afficher la version de Java installée avec laquelle Windows fonctionne (cela n'a pas d'importance pour nous) :

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

0.6 Installations optionnelles

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

- Miktek pour produire des documents PDF avec Latex
- Rtools pour développer des packages **R** et compiler le code.

0.7 Problèmes que vous pouvez rencontrer

0.7.1 Problèmes lors de l'installation des packages **R

Si vous obtenez l'erreur suivante lors de l'installation des packages **R** :

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

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

Le problème ne vient pas de Java mais du package **R**. Par défaut, le package est installé à partir d'un fichier "source", ce qui signifie que le package est recompilé. Pour des raisons informatiques, lors de la compilation par défaut, **R** utilise les paramètres du système (Windows) (qui n'a pas forcément la bonne version de Java).

Il y a deux solutions :

- Compiler le package en l'installant à partir du fichier binaire :

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

- Spécifier que nous voulons utiliser les paramètres locaux :

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

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

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

Le package **{RJDemetra nécessite la version 8 ou supérieure de Java pour fonctionner. Si un autre package a été chargé avant **{RJDemetra}** via la fonction `library()` et qui ne nécessite pas une version Java à jour, alors une ancienne version Java sera utilisée pendant toute la session (**R** est réfractaire au changement de version en cours de session). Dans le cas de l'utilisation de **{RJDemetra}** dans un programme, il faut spécifier au tout début du programme que **R** doit utiliser la version 8 de Java, via la commande :

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

ou chargez d'abord **{RJDemetra}**

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

Sinon il vous faut redémarrer une nouvelle session **R**

0.7.3 `Error array index = -1`

Le message du type `Error array index = -1` indique qu'une variable auxiliaire n'a pas été trouvée. Il peut s'agir d'un régresseur de calendrier ou d'autres variables définies par l'utilisateur (effet de Pâques, PSO = pure seasonal outlier…).

0.7.4 La fonction `cruncher_and_param(...)` du package {JDCruncheR} renvoie une erreur

Lorsque vous utilisez la fonction `cruncher_and_param(...)` du package **{JDCruncheR}**, vous pouvez obtenir l'erreur suivante :

```
## Error in eval(expr, envir, enclos): Error in cruncher(workspace = workspace, cruncher_bin_directory =
##   There is an error in the path to the cruncher bin folder
```

Cela signifie que le chemin d'accès à l'outil de calcul est mal configuré. Pour résoudre ce problème, vous devez spécifier à R le chemin d'accès à l'outil de calcul au début du programme à l'aide de la fonction `options(...)` :

```
options(cruncher_bin_directory = "C:/Users/Software/jwsacruncher-2.2.4-bin/bin")
```

Pour vérifier que le chemin est valide, vous pouvez utiliser la fonction `getOption(...)` :

```
getOption("cruncher_bin_directory")
```