


Configuring JDemetra+ and related R tools

Tanguy BARTHELEMY

Context

This document complements the [quick installation guide] (https://github.com/TanguyBarthelemy/JD-Tutorials/blob/master/Sheets/JDemetra+_quick_installation_guide_EN.pdf) by providing more details on the configuration required to run JDemetra+ and related tools.

On computers without administrator rights (*professional computer, for example*), it is recommended to create a folder **Software** under `C:\Users\...\Software` or directly under `C:\Users\Software` where all software will be installed.

 Warning: when we specify an **absolute** path for a software (JDemetra+, Java, **R**, ...) in a program, a shortcut, a variable, ..., it must be modified each time any root repository is moved.

0.1 Installation of JDemetra+ Graphical user interface

To install JDemetra+ Graphical User Interface see the [quick installation guide] (https://github.com/TanguyBarthelemy/JD-Tutorials/blob/master/Sheets/JDemetra+_quick_installation_guide_EN.pdf)

0.2 Installation of the cruncher

To install JDemetra+ cruncher see the [quick installation guide] (https://github.com/TanguyBarthelemy/JD-Tutorials/blob/master/Sheets/JDemetra+_quick_installation_guide_EN.pdf)

0.3 Installation of R and Rstudio

The JDemetra+ algorithms can be accessed in **R** via **R** packages. To use **R**, it is better to use an IDE like Rstudio. All the executable files to download are under <https://posit.co/download/rstudio-desktop/#download>.

0.3.1 Installation of R

To install **R**, you should:

- **Download** the binary file `R-4.3.2-win.exe` under <https://cran.rstudio.com/bin/windows/base/>
- **Execute** the executable to parameter and install **R**.

0.3.2 Installation of Rstudio

Download the last Rstudio version (under <https://posit.co/download/rstudio-desktop/#download>) and the **installer**.

If the installation via the file `.exe` fails (because it requires higher rights (administrator, elevation, ...), we will get a portable version of the Software. To do this:

- **Download** and **unzip** the compressed folder .zip in a folder named “Rstudio” (under C:\Users\Software)
- **Create a shortcut** of the file rstudio.exe on the Desktop.

0.3.3 Installation of R packages

To install a R package, there are several methods:

- Either it is on CRAN and you can install it directly with the function `install.packages()`
- Or it is on Github and you can install it directly with the function `install_github()` from the package **remotes**
- Or you have to retrieve the package from a folder (binary format) (.zip) and then install it with the function `install.packages()` with the arguments `repos = NULL`, `type = "binary"`.

0.3.3.1 R packages for JDemetra+ version 2.2.4

The packages in version 2 are:

Name	Available on CRAN	Available on AUS	Github link
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

The packages installation code is below:

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

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

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

# Under AUS and on Insee computers
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 R packages for JDemetra+ version 3.x

Currently version 3 packages are NOT on CRAN.

To install them, you need to go through Github:

```
# If remotes is not installed
# 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/rjd3demetra3")

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")
```

0.4 Environment variables

0.4.1 In Rstudio

To add environment variables in Rstudio, you need to add the variable's name and the variable's value. There are 2 ways to fill it:

- Using the `.Renviron` file:
 - **Launch** the code `file.edit("~/Renviron")` or `usethis::edit_r_environ()` (with the package `usethis` and the argument `scope` which equals `"user"` or `"project"` if you are in a R project)
 - **Add** the variables to the file (with new lines)
 - **Save** the file
- Using the `.Rprofile` file:
 - **Launch** the code `file.edit("~/Rprofile")` or `usethis::edit_r_profile()` (with the package `usethis` and the argument `scope` which equals `"user"` or `"project"` if you are in a R project)
 - **Add** the variables with the fonction `Sys.setenv()` (with new lines)
 - **Save** the file

0.4.2 PATH

The environment variable `PATH` in **R** is used to indicate to **R** where to find the executable files.

When you install a new software (for example JDemetra+, Rtools, Java...) that Rstudio uses, you have to modify this environment variable:

- **Get** the actual value of the variable `PATH` via the **R** command `Sys.getenv("PATH")` (Rstudio returns a succession of addresses as `C:/WINDOWS/system32;C:/WINDOWS`)

- **Copy paste** this value after `PATH =` and add the paths towards the folder `\bin\` (binary) of the software newly installed, by separating them with semicolon (without space before or after).

For the Rtools installation, the path is `C:\rtools42\mingw64\bin` (depending on where Rtools was installed). You have to add `C:\rtools42\mingw64\bin` or `C:/rtools42/mingw64/bin` (In **R**, `\` is a special character, so you have to replace the `\` by `/` or by `\\`). The path becomes `C:/WINDOWS/system32;C:/WINDOWS;C:/rtools42/mingw64/bin`.

- **Modify** the variable with the function `Sys.setenv()`. For Rtools, the command to launch is:

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

i NB: Generally a 32 bits version and a 64 bits version are available for downloading and installing a software. You need to check your processor type of your operating system to choose the right folder to download.

For this, you can launch the command:

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

According to the result, the version is 32 bits or 64 bits :

Version	Output
64 bits	/x64 x86_64
32 bits	/i386 x86_32

More information on the variable `PATH` via the page <https://java.com/fr/download/help/path.xml>.

0.5 Check your set up

To ensure that everything works fine, you can launch some example of **R** code and check that there is no error:

```
library("RJDemetra")

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

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

To check the Java version we are using on **R**, you can try to install and use the package **rJava** and launch the command below:

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

If the installation of **rJava** returns an error, it means that Java was incorrectly installed or incorrectly configured on **R**. You need to get back to the section Environment variables.

This block of code tests the Java version with which **R** works:

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

Finally, you can display the Java version installed with which Windows works (it doesn't matter to us):

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

0.6 Optional installations

Some supplementary installations are optional (that is they are no mandatory but bring external features):

- Miktek to produce PDF document with Latex
- Rtools to develop **R** packages and compile the code

0.7 Problems you may encounter

0.7.1 Problems installing **R** packages

If you get the following error while installing **R** packages:

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

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

The problem doesn't come from Java but from the **R** package. By default, the package is installed from a "source" file, it means that the package is recompiled. For some computing reasons, when compiling by default, **R** uses the system (Windows) parameters (which doesn't have necessarily have the correct Java version).

There are two solutions:

- Compile the package by installing from the binary file:

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

- Specify that we want to use the local parameters:

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

 More information: <https://github.com/jdemetra/rjdemetra/wiki/Installation-manual>

0.7.2 The command `library("RJDemetra")` returns an error message

The package **RJDemetra** requires Java version 8 or higher to work. If another package has been loaded before **RJDemetra** via the function `library()` and which doesn't require an updated Java version, then an old Java version will be used during all the session (**R** is refractory to in-session version change). In case of using **RJDemetra** in a program, you have to specify at the very beginning of the program that **R** must use Java version 8, via the command:

```
# Where Java is installed
Sys.setenv(JAVA_HOME = "C:/Users/Software/Java17/jdk17")
```

or load **RJDemetra** first

```
# At the beginning of program
library("RJDemetra")
```

Otherwise you have to restart a new **R** session.

0.7.3 Error array index = -1

The message of the type `Error array index = -1` tells that an auxiliary variable is not found. It can be calendar regressor or other user defined variables (Easter effect, PSO = pure seasonal outlier...).

0.7.4 The function `cruncher_and_param(...)` of the **JDCruncherR** package returns an error

When you use the function `cruncher_and_param(...)` of the **JDCruncherR** package, you can get the following error:

```
## 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
```

That means that the path to the cruncher is incorrectly configured. To solve this, you need to specify to **R** the path to the cruncher at the start of the program with the function `options(...)`:

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

To check that the path is valid, you can use the function `getOption(...)`:

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