The rjdworkspace package

Re-organising series between workspaces

November 2021

Alice Tchang

Insee, DMS

Abstract: JDemetra+ (JD+) is the tool recommended by Eurostat to seasonally adjust time series. An external access to its Java routines is possible via R thanks to an ecosystem of packages, many of which are based on the RJDemetra package. rjdworkspace, the package subject of this note, is no exception. While RJDemetra offers access to the JD+ algorithms and diagnostics, rjdworkspace operates one level above and enables the user to move series from a workspace or a multiprocessing to another (along with its complete metadata), as well as to change its raw data, its specification and even the path to the raw data file in the xml file. These can prove particularly useful during annual reviews, studies or the production process. In this note, we will see the main functions of the rjdworkspace package, as well as how and when to use them.

Keywords: Time Series, R, Seasonal Adjustment, JDemetra+, RJDemetra, rjdworkspace

JEL Classification : C01, C22

Table of contents

Introduction

In order to use the JDemetra+ interface (GUI), the data must be stored into files called "workspaces". These can host one or several groups of series, called "SAProcessings" (SAP) or "multi-processings". While RJDemetra enables the user to customise seasonal adjustment parameters, the rjdworkspace package is meant to help manipulate the objects 'series' (or *sa_item*) themselves and their metadata.

Indeed, in R, an *sa_item* is composed of three types of elements: a time series, a specification (model parameters) and metadata. The metadata encompasses a variety of information on the raw series file (the path qhere it is stored, its extension (txt, xls), the date format) as well as information on series' frequency, their position in the SAP and the comments.

Some operations that can't currently be done easily under JD+ or with the RJDemetra package include:

- merging two workspaces into a new workspace
- changing the raw time series of a 'series' object
- changing a series' global specification
- changing the path to the raw data file
- updating a workspace's metadata qu'est-ce qu'il y a de plus dans les metadonnées, que le chemin vers les d brutes? le nom de la série, son rang dans le ws, le type de fichier contenant les brutes (extension), le span...?

These features would be useful:

- during annual reviews, when merging the workspaces containing the currently used models and the automatically estimated models. While the RJDemetra package is of great use to retrieve diagnostics, to produce custom quality reports and comparison tables, the user then has to manually merge the workspaces into the final one, which can be quite tedious (replicating the changes into the final workspace for each series).
- during studies, to quickly apply and test specifications on series, as well as to split and merge workspaces to regroup series according to tests results
- during collaborative work when all parties don't have access to the same disk partition: the path to the raw series must be changed throughout the workspace if it is to be crunched. Also, leaving information on the changes made on or problem encountered with a specific series could be handy.

Main functionalities

Therefore, some of the main rjdworkspace functionalities include:

- a customisable left join function and a modified "union all" function to merge two workspaces;
- At the series' level, functions include adding and removing series from a multiprocessing, as well as changing a series' raw data while keeping its original specification (model) and vice versa;
- Metadata-wise, the package contains a function to update a whole workspace's metadata with another one's, as well as functions to only change the path to the raw data file, rename series and retrieve and update the comments of a series, an SAP or a workspace.

Installation procedure

The package can be downloaded from the Insee Lab github: `https://github.com/InseeFrLab/rjdworkspace`.

From a technical standpoint: since RJDemetra relies on the rJava package that requires at least Java SE 8, rjdworkspace also requires at least that version.

1 Functions that operate on one workspace

The rjdworkspace offers functions to manipulate sa_items, whether it be **adding**, **replacing** or **removing them**, as well as to **change several characteristics**: a series' name, raw time series, specification and associated comment.

1.1 Adding or replacing a single series in a workspace

The functions

add_new_sa_item adds the sa_item at the end of the specified SAP.

replace_sa_item "slides" a series into the specified SAP at the specified position.

pour les deux fonctions: est-ce que les variables externes / usedefined sont transférées? (notamment add_new_sa_item vs add_sa_item)

Examples

```
# To add a series at the end of a SAP
add_new_sa_item(SAProcessing, sa_item)

# To swap a series for another in a SAP
replace_sa_item(SAProcessing, series_position, sa_item)
```

1.2 Removing one or several series from a SAProcessing

The functions

remove_sa_item deletes the series in the specified position from the SAP. When called only on a SAP, it deletes its first series.

remove_all_sa_item clears a whole SAP.

Examples

```
# To delete one series from a SAProcessing
remove_sa_item(SAP, series_position)

# Or all
remove_all_sa_item(SAProcessing)
```

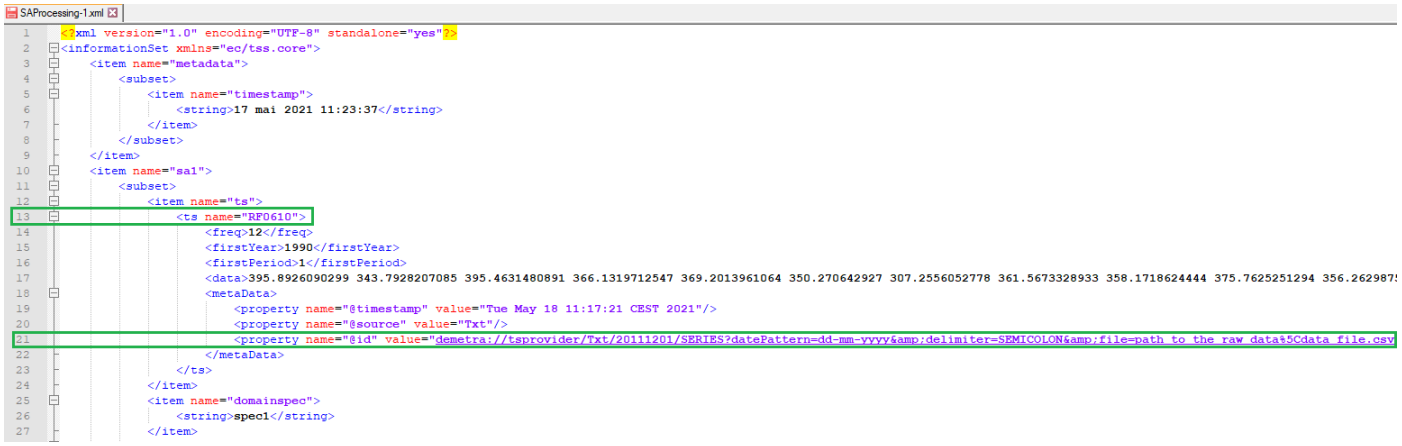
1.3 Changing the path to the raw data file

The function

update_path changes the path to the raw data file for all series of a workspace.

Technical details:

- before beginning the update process, the function checks that all workspace's series are present in the raw data file indicated by the path argument. Therefore, said path must effectively lead to a data file. If some series aren't found, their list is returned and the function updates the path for all other series;
- said path must be entered in the classical "R format": translation of the special characters "\", "/", " " and ":" into their ASCII counterpart is done by the function;
- this function is set up for csv files with one single sheet per file. It could easily be customised for multi-sheet files such as Excel spreadsheets, but since we've observed the Excel date format can be unstable, we don't recommend using it and haven't coded the adapted function; for now, the function only operates on workspaces that contain one SAProcessing;
- the parameter "param" refers to the number of lines that separate the one containing the series name and the one containing the file path (8 by default):



Example

```
update_path(ws, "new_path/new_file.csv")

# customisation of the spacing parameter
update_path(ws, "new_path\\new_file.csv", param = 5)
```

1.4 Changing a series' name, raw time series, specification, or comment

To modify a series' name, raw time series, specification or associated comment, one must operate in two steps:

- apply the appropriate "setter" function to an sa_item;
- store the new (and updated) sa_item into a workspace, by either replacing a pre-existing series (function replace_sa_item) or adding it into an SAP (function add_new_sa_item).

The functions

They have the classical "setter" names: **set_name**, **set_ts**, **set_spec**, **set_comment** (and **get_comment**).

Technical details:

- **set_ts** changes the raw data stored in the xml file but does **not** update the path to the new data source;
- **set_spec** doesn't take user-defined variables into account;
- **get_comment** can be used to retrieve comments attached to a series, a SAProcessing or a workspace. When more than a comment is retrieved, the output of the function is a vector (or a vector of vectors) of comments.

Functions syntax

set_ts(sa_item, new_time_series)

set_name(sa_item, "new_name")

set_spec(sa_item, spec_object)

get_comment(sa_item)
set_comment(sa_item, "write your comment here")

Examples

```
# Loading a workspace
ws <- load_workspace("path_to_workspace/my_ws.xml")
compute(ws)

# Extracting its SAProcessings
```

```
SAP <- get_all_objects(ws)

# And the first SAP's series
series <- get_all_objects(SAP[[1]])

# To operate on the first SAP's third series:
my_series <- series[[3]]

# To change its raw data for the AirPassengers series
my_series <- set_ts(my_series, AirPassengers)

# To retrieve a specification and apply it to the series
sa_x13 <- jx13(ipi_c_eu[, "FR"])
my_series <- set_spec(my_series, sa_x13)

# To rename the series
set_name(my_series, "series_with_x13")

# To add a comment to document the change
set_comment(my_series, "x13 IPI model applied and name changed")

# And to store the series back in its place in the original workspace (as a series update)
replace_sa_item(SAP, 3, my_series)

# Or to add it at the end of an SAP in another workspace (or at the end of the same workspace)
add_new_sa_item(other_SAP, my_series)

# Don't forget to save the resulting workspace!
save_workspace(ws, "path_to_folder/updated_workspace.xml")
```

2 Functions that operate on two workspaces

These functions industrialise the transfer of metadata or series from one workspace to another, whether they be already in the original workspace but in a different version (for example with a different specification) or not.

2.1 Metadata update

A series' metadata include the path to the raw data file, its extension (csv, xls), the series frequency, etc.

It can be done either **on a single series**, via the function set_metadata, or **on a whole workspace** thanks to the functions update_metadata and update_metadata_roughly.

set_metadata enables to change one series' metadata in bulk

update_metadata is the generalisation of set_metadata to a whole workspace. Controls are made that both workspaces' structure is the same before updating the second workspace's metadata to the first workspace. The replacement is done for all series in common between both workspaces.

update_metadata_roughly is a simpler version that doesn't perform unicity or coherency checks (including series order and names) before updating the metadata. Instead, it assumes that both workspaces have the same structure and transfers the metadata series by series: the transfer is done between the workspaces' first series of the first SAP, then between the second series of the first SAP and so on.

Examples

```
# To apply the specifications and metadata of a number of series to others,
# or even from a whole workspace to another one (for example, when comparing a workspace
# of reference to a test one):
updated_ws <- update_metadata(ws_reference, ws_test)
save_workspace(updated_ws, path_final_ws.xml)
```

2.2 Specification update

The function **replace_series()** replaces a number of series in a given workspace (ws1), by the same-named series from another workspace (ws2). Therefore, the function modifies the first workspace: it doesn't create a new merged workspace.

Technical details:

- if the second workspace contains external variables such as trading days regressors or intervention variables, they must be imported into the first workspace before calling replace_series;
- by default, the SAPs used are the first on each workspace. If a SAP name is specified, it must correspond to the SAPs used in both workspaces.
- calls of this function are cumulative: if a series *series*1 is replaced in a workspace ws1, and then a series *series*2 is, both series will be replaced in ws1. If the final decision is to only replace *series*2, the workspace ws1 must be re-loaded and re-computed before using the function replace_series with *series*2 as the "selected_series" argument.

Examples

```
# To replace a selection of series by the same series with a different specification
# (from another workspace)
ws1 <- replace_series(ws1, ws2, selected_series, "SAProcessing-1", print_indications = TRUE)
# If the SAP isn't specified, the first SAP of each workspace will be used
ws1 <- replace_series(ws1, ws2, series_to_replace)
```

2.3 Workspace completion

The function **transfer_series()** enables the user to augment a SAProcessing with series from another SAProcessing, that **are not** already in the first SAProcessing. It can be used to regroup several SAPs (or mono-SAP-workspaces) into one SAP (workspace). The possible deletion of the transfered SAPs (workspaces) isn't taken in charge and must be done by hand through the JD+ interface (in the appropriate folder).

Technical details:

- if the second workspace contains external variables such as trading days regressors or intervention variables, they must be imported into the first workspace before calling transfer_series();
- by default, the SAPs used are the first of each workspace. If only one SAP name is specified, it will be used for the first workspace. If both are, they will be used in the order they are entered.

NB. In the case of a SAP reunion into the first workspace, the function doesn't delete the SAP from the second workspaces (nor the transfered series): such operation must be done separately via the interface.

Examples

```
# Here, the SAPs can have different names in each workspace
augmented_ws1 <- transfer_series(ws1, ws2, "mp_to", "mp_from", print_indications = TRUE)

# Series will be transfered from ws2's first SAP
augmented_ws1 <- transfer_series(ws1, ws2, "mp_to", print_indications = TRUE)
```

3 When to use and not to use the rjdworkspace package

There are two ways to create and modify a workspace: via the JD+ interface and directly in R, using the RJDemetra package. Both enable the user to perform seasonal adjustment with both Tramo-Seats and the X13-Arima method, starting from scratch or from a pre-defined (or a user-defined) specification, including user-defined calendar regressors and intervention variables.

However, **the metadata contained in the xml file of the SAProcessing folder isn't the same with both methods**. Indeed, after importing raw data into JD+ and dropping them into a SAProcessing

panel, the resulting workspace's metadata contains the path to said raw data file. When creating a workspace from scratch with RJDemetra, the series imported in R are correctly saved in the xml file (the raw values can be found in the file for each series) but the link to the original data file isn't. Therefore, functions like *replace_series*() will work but the resulting workspace will not be usable in recurrent production because it will not be "crunchable".

In the same fashion, some of the "setters" functions can be used during a one-shot study or test but not in a recurrent process such as an annual review because they update a sa_item's component (raw time series) but not its metadata.

Conclusion

Used in conjunction with RJDemetra, the rjdworkspace package can be very useful both for production as well as for exploratory study purposes.

When working on a one-shot project where the cruncher will not be needed, workspaces can be created either via the JD+ interface or in R with RJDemetra functions. However, workspaces for recurrent use (typically monthly or quarterly productions), must be generated with JDemetra+ so that the paths to the raw data files are stored in the metadata.

rjdworkspace is complementary to RJDemetra in the sense that it enables the user to mix and match parts of a "series" sa_item object (the time series, the specification and comments), as well as series between SAProcessings and workspaces.

Therefore, the rjdworkspace functions can both be used for massive automated configuration as well as fine tuning specific series: it is the logical extension of the RJDemetra package.