Package ‘rjdworkspace’

January 21, 2025

Type Package

Title Manipulate 'JDemetra+' Workspaces

Version 1.1.9

Description Set of tools to manipulate the 'JDemetra+' workspaces.
Based on the 'RJDemetra' package (which interfaces with version 2 of the 'JDemetra+' (<https://github.com/jdemetra/jdemetra-app>), the seasonal adjustment software officially recommended to the members of the European Statistical System (ESS) and the European System of Central Banks).
This package provides access to additional workspace manipulation functions such as metadata manipulation, raw paths and wrangling of several workspaces simultaneously.
These additional functionalities are useful as part of a CVS data production chain.

Depends R (>= 3.1.1)

Imports rJava (>= 0.9-8), RJDemetra, XML

License EUPL

URL https://github.com/InseeFrLab/rjdworkspace

BugReports https://github.com/InseeFrLab/rjdworkspace/issues

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Tanguy Barthelemy [aut, cre, art],
Alain Quartier-la-Tente [aut] (<https://orcid.org/0000-0001-7890-3857>),
Institut national de la statistique et des études économiques [cph]
(https://www.insee.fr/),
Anna Smyk [aut]

Maintainer Tanguy Barthelemy <tanguy.barthelemy@insee.fr>

Repository CRAN

Date/Publication 2025-01-21 13:20:01 UTC

Contents

`copy_ws` *Copy a WS*

Description

Copy a WS

Usage

```
copy_ws(ws_name, from, to = tempdir(), overwrite = TRUE, verbose = TRUE)
```

Arguments

`ws_name`	the name of the WS
`from`	the path to the folder containing the WS (the XML file + the WS folder)
`to`	the path to the new folder which will contains the WS (the XML file + the WS folder)
`overwrite`	Overwrite existing file (Defaults to TRUE)
`verbose`	A boolean to print indications on the processing status (optional and TRUE by default)

Value

the function returns invisibly (with `invisible()`) a boolean specifying if the transfer was done or an error if the specified paths or workspace don't exists

Examples

```
# Déplacement d'un WS dans un environnement temporaire
destination_dir <- tempdir()

# Copy of a worspace in a temporary environment
copy_ws(
  ws_name = "ws_output",
  from = file.path(system.file("extdata", package = "rjdworkspace"), "WS"),
  to = destination_dir
)
```

`get_comment` *Extract comments*

Description

Function to extract the comments of a workspace

Usage

```
get_comment(x)
```

Arguments

`x`	the object from which the comments are retrieved.

Value

A string or list of string with all the comment contained in a SA-Item, a SA-Processing or a workspace (depending on the argument `x`).

Examples

```
library("RJDemetra")
ws_dir <- file.path(system.file("extdata", package = "rjdworkspace"), "WS")

path_ws_to <- file.path(ws_dir, "ws_output.xml")

ws_output <- load_workspace(path_ws_to)
print(get_comment(ws_output))

sap_output <- get_object(ws_output, pos = 3)
print(get_comment(sap_output))

sa_item <- get_object(sap_output, pos = 3)
print(get_comment(sa_item))
```

manipulate_sa_item *Manipulate SaItems*

Description

Functions to remove/replace/add a `sa_item` from/to a SA-Processing.

Usage

```
remove_sa_item(sap, pos = 1L)

remove_all_sa_item(sap)

replace_sa_item(sap, pos = 1L, sa_item)

add_new_sa_item(sap, sa_item)
```

Arguments

`sap`	the SA-Processing.
`pos`	the index of the `sa_item` to remove or to replace.
`sa_item`	`sa_item` object.

Value

The functions `remove_sa_item()`, `remove_all_sa_item()` and `replace_sa_item()` return invisibly (with `invisible()`) `TRUE` or an error. The function `add_new_sa_item()` returns invisibly (with `invisible()`) the updated `SA-Item`.

Examples

```
library("RJDemetra")

sa_x13 <- jx13(series = ipi_c_eu[, "FR"])
sa_ts <- jtramoseats(series = ipi_c_eu[, "FR"])

wk <- new_workspace()
sap1 <- new_multiprocessing(workspace = wk, name = "sap-1")
add_sa_item(workspace = wk, multiprocessing = "sap-1",
            sa_obj = sa_x13, name = "X13")
add_sa_item(workspace = wk, multiprocessing = "sap-1",
            sa_obj = sa_ts, name = "TramoSeats")

sa_item1 <- get_object(x = sap1, pos = 1L)

remove_sa_item(sap = sap1, pos = 1L) # Remove the first sa-item
add_new_sa_item(sap = sap1, sa_item = sa_item1) # Add the sa-item at the end
```

```
# To replace the first sa_item by "sa_item1"
replace_sa_item(sap = sap1, pos = 1L, sa_item = sa_item1)
```

`replace_series` *Partial update of a workspace metadata*

Description

`replace_series()` allows to update a selection of series by the same-named series from another workspace. When only the metadata differs, it is the partial version of the update_metadata function.

Generic function to identify and return the duplicates in a list

Usage

```
replace_series(
  ws_from,
  ws_to,
  selected_series,
  mp_from_name,
  mp_to_name,
  verbose = TRUE
)

verif_duplicates(s)

verif_ws_duplicates(ws, verbose = TRUE)
```

Arguments

`ws_from`	The workspace containing the most up-to-date version of the selected_series series
`ws_to`	The workspace to update
`selected_series`	The vector containing the series-to-update's names.
`mp_from_name`	The name of the SA-Processing containing the series to update (optional)
`mp_to_name`	The name of the SA-Processing to update (optional)
`verbose`	A boolean to print indications on the processing status (optional and TRUE by default)
`s`	a list of characters
`ws`	The workspace to scan

Details

If the arguments `mp_from_name` & `mp_to_name` are unspecified, the update will be performed using the workspaces' first SAProcessing. If a series is specified in the selected_series vector is missing in a workspace, no replacement will be performed and the function will return the list of missing series. Otherwise, if all is well, the function returns the workspace ws_to updated.

`verif_duplicates()` identifies and returns the duplicates in a list `verif_ws_duplicates()` identifies duplicated series in a SAProcessing (SAP) and SAProcessings in a workspace

Value

the updated `workspace`

If there are no duplicates, the function returns an empty data frame. Otherwise, it returns a data frame giving the name and number of duplicates found within the argument (list).

a list containing the name and number of occurences of duplicated SAPs and series

Examples

```
library("RJDemetra")
dir_ws <- tempdir()
template_ws <- file.path(system.file("extdata", package = "rjdworkspace"),
                         "WS")
# Moving the WS in a temporary environment
copy_ws(
    ws_name = "ws_output",
    from = template_ws,
    to = dir_ws
)
copy_ws(
    ws_name = "ws_input",
    from = template_ws,
    to = dir_ws
)
path_ws_from <- file.path(dir_ws, "ws_input.xml")
path_ws_to <- file.path(dir_ws, "ws_output.xml")
ws_input <- load_workspace(path_ws_from)
ws_output <- load_workspace(path_ws_to)

replace_series(
    ws_from = ws_input,
    ws_to = ws_output,
    mp_from_name = "SAProcessing-2",
    mp_to_name = "SAProcessing-2",
    selected_series = c("RF1039", "RF1041"),
    verbose = TRUE
)


s <- c("a", "b", "a", "c", "a", "c")
print(rjdworkspace:::verif_duplicates(s))
```

`set_comment` *Change comment*

Description

Function to change the comments of a sa_item object

Usage

```
set_comment(x, comment)
```

Arguments

`x`	the sa_item of which the comments will be changed.
`comment`	the new comment.

Value

a new sa_item.

`set_metadata` *Set the metadata of a SaItem*

Description

Function to set the name of a `"sa_item"` from the one contained in another `"sa_item"`.

Usage

```
set_metadata(sa_from, sa_to)
```

Arguments

`sa_from`	the `"sa_item"` object from which the desired metadata is retrieved.
`sa_to`	the `"sa_item"` object to modify.

Value

a new `"sa_item"` with the specification of `sa_to` and the metadata of `sa_from`.

set_name *Set the name of a SaItem*

Description

Function to set the name of a `"sa_item"`.

Usage

```
set_name(sa_item, name)
```

Arguments

`sa_item`	a `"sa_item"` object.
`name`	the new name.

Value

a new `"sa_item"` with the new name.

Examples

```
library("RJDemetra")

sa_x13 <- jx13(series = ipi_c_eu[, "FR"])

wk <- new_workspace()
sap1 <- new_multiprocessing(workspace = wk, name = "sap-1")

add_sa_item(workspace = wk, multiprocessing = "sap-1",
            sa_obj = sa_x13, name = "Wrong name")

sa_item1 <- get_object(x = sap1, pos = 1L)

new_sa_item <- set_name(sa_item = sa_item1, name = "Good name")
replace_sa_item(sap = sap1, pos = 1L, sa_item = new_sa_item)

# The first sa_item of sap1 is now "Good name"
get_name(x = get_object(x = sap1, pos = 1L))
```

`set_spec` *Set the specification of a SaItem*

Description

Function to set the specification of a `"sa_item"`.

Usage

```
set_spec(sa_item, spec)
```

Arguments

`sa_item`	a `"sa_item"` object.
`spec`	the object into which the new specification is extracted/stored.

Value

a new `"sa_item"` with the new specification

Examples

```
library("RJDemetra")

sa_x13 <- jx13(series = ipi_c_eu[, "FR"])
sa_ts <- jtramoseats(series = ipi_c_eu[, "FR"])

wk <- new_workspace()
sap1 <- new_multiprocessing(workspace = wk, name = "sap-1")

add_sa_item(
    workspace = wk,
    multiprocessing = "sap-1",
    sa_obj = sa_x13,
    name = "tramo seats"
)

sa_item1 <- get_object(x = sap1, pos = 1L)
new_sa_item <- set_spec(sa_item = sa_item1, spec = sa_ts)

# The first sa_item is now seasonally adjusted with TRAMO-SEATS
replace_sa_item(sap = sap1, pos = 1, sa_item = new_sa_item)
```

set_ts *Change the input time series of a SaItem*

Description

Function to change the input time series of a SaItem

Usage

```
set_ts(sa_item, ts)
```

Arguments

`sa_item`	the `sa_item` to modify.
`ts`	the new `stats::ts()` object.

Value

a `sa_item`

Examples

```
library("RJDemetra")

# Definition of the original time series
sa_x13 <- jx13(series = ipi_c_eu[, "FR"])

wk <- new_workspace()
sap1 <- new_multiprocessing(workspace = wk, name = "sap-1")

# Adding a new SA-Item (`sa_x13`) to `sap1`
add_sa_item(workspace = wk, multiprocessing = "sap-1",
            sa_obj = sa_x13, name = "X13")

# Retrieving the adjusted series
sa_item1 <- get_object(x = sap1, pos = 1L)

# Creation of a new sa_item and change of the input time series
new_sa_item <- set_ts(sa_item = sa_item1, ts = ipi_c_eu[, "BE"])
# Replacement of the series in the workspace
replace_sa_item(sap = sap1, pos = 1L, sa_item = new_sa_item)
```

transfer_series *Transfer_series*

Description

To copy & paste series from one workspace to another

Usage

```
transfer_series(
  ws_from,
  ws_to,
  selected_series,
  pos_sap_from,
  pos_sap_to,
  name_sap_from,
  name_sap_to,
  verbose = TRUE,
  create_sap = TRUE,
  replace_series = TRUE
)
```

Arguments

`ws_from`	The workspace containing the additionnal series
`ws_to`	The workspace to add series to
`selected_series`	The vector containing the series-to-update's names.
`pos_sap_from`	The position of the SA-Processing to transfer the series from
`pos_sap_to`	The position of the SA-Processing to transfer the series to
`name_sap_from`	The name of the SA-Processing to transfer the series from (optional)
`name_sap_to`	The name of the SA-Processing to transfer the series to (optional)
`verbose`	A boolean to print indications on the processing status (optional and TRUE by default)
`create_sap`	A boolean to create a new SA-Processing if not existing (optional)
`replace_series`	A boolean to replace existing series (optional)

Details

To use this function you need to first launch `load_workspace` and after `save_workspace` to save the changes.

`name_sap_to` and `name_sap_from` refer to the SAP's name and not SAP's file's name.

The transfer will fail if: - `name_sap_from` doesn't exist - `pos_sap_from` < 0 or exceed the maximum number of SAP - `pos_sap_to` < 0 or exceed the maximum number of SAP - The arguments pos_sap_from and name_sap_from are refering to differents objects. - The arguments pos_sap_to and name_sap_to are refering to differents objects.

If name_sap_to and pos_sap_to are unspecified, the update will be performed using the workspaces' first SAProcessing (same for the SAP from). However if the informations of one on the two SAP (from or to) are specified (name or position), they will be attributed by default to the other worskpace.

If name_sap_to doesn't refer to an existing SAP, a new SAP will be created (if create_sap is TRUE).

If a sa_item has a specification which uses external regressor, you have to be sure that the regressors are also in the destination workspace.

Value

the workspace ws_to augmented with series present in ws_from and not already in ws_to

Examples

```
library("RJDemetra")
dir_ws <- tempdir()
template_ws <- file.path(system.file("extdata", package = "rjdworkspace"),
                          "WS")
# Moving the WS in a temporary environment
copy_ws(
    ws_name = "ws_output",
    from = template_ws,
    to = dir_ws
)
copy_ws(
    ws_name = "ws_input",
    from = template_ws,
    to = dir_ws
)
path_ws_from <- file.path(dir_ws, "ws_input.xml")
path_ws_to <- file.path(dir_ws, "ws_output.xml")
ws_input <- load_workspace(path_ws_from)
ws_output <- load_workspace(path_ws_to)

# Existing SAP
transfer_series(
    ws_from = ws_input,
    ws_to = ws_output,
    name_sap_from = "SAProcessing-1",
    name_sap_to = "SAProcessing-1",
    verbose = TRUE
)

transfer_series(
    ws_from = ws_input,
    ws_to = ws_output,
    pos_sap_from = 1,
    pos_sap_to = 1,
    verbose = TRUE
)

# Existing series
transfer_series(
    ws_from = ws_input, ws_to = ws_output,
    pos_sap_from = 2,
    pos_sap_to = 2,
    verbose = TRUE,
    replace_series = FALSE
)
transfer_series(
    ws_from = ws_input, ws_to = ws_output,
    pos_sap_from = 2,
    pos_sap_to = 2,
    verbose = TRUE,
    replace_series = TRUE
)

# Create a new SAP
# transfer_series(ws_from = ws_input, ws_to = ws_output,
#                 name_sap_from = "SAProcessing-1",
#                 name_sap_to = "New-SAProcessing-from-R",
#                 verbose = TRUE,
#                 create = FALSE)

transfer_series(
    ws_from = ws_input, ws_to = ws_output,
    name_sap_from = "SAProcessing-1",
    name_sap_to = "New-SAProcessing-from-R",
    verbose = TRUE,
    create = TRUE
)

RJDemetra::save_workspace(workspace = ws_output, file = path_ws_to)
```

`update_metadata` *Update the metadata from a workspace*

Description

Functions to update the metadata of a workspace by those contained in another one

Usage

```
update_metadata(ws_from, ws_to)

update_metadata_roughly(ws_from, ws_to)
```

Arguments

`ws_from`	Workspace that contains the new metadata.
`ws_to`	Workspace to update.

Details

`update_metadata()` checks the SA-Processings and SaItems' names within the two workspaces before updating ws_to's metadata. `update_metadata_roughly()` does not do any checks: `ws_to`'s first SA-Processing's first SaItem metadata is updated with `ws_from`'s first SA-Processing's first SaItem metadata. Both functions create and return a new workspace containing the updated series.

Value

the updated `workspace`

Examples

```
library("RJDemetra")

path_to_ws1 <- file.path(
    system.file("extdata", package = "rjdworkspace"),
    "WS/ws_example_1.xml"
)
path_to_ws2 <- file.path(
    system.file("extdata", package = "rjdworkspace"),
    "WS/ws_example_2.xml"
)

ws_1 <- load_workspace(path_to_ws1)
compute(ws_1)
ws_2 <- load_workspace(path_to_ws2)
compute(ws_2)

updated_workspace <- update_metadata_roughly(ws_from = ws_1, ws_to = ws_2)
path_to_output <- file.path(tempdir(), "ws_update_meta_roughly.xml")
save_workspace(workspace = updated_workspace, file = path_to_output)

updated_workspace <- update_metadata(ws_from = ws_1, ws_to = ws_2)
path_to_output <- file.path(tempdir(), "ws_update_meta.xml")
save_workspace(workspace = updated_workspace, file = path_to_output)
```

`update_path` *Update the path to the raw series file*

Description

Function to update the path of the raw data file in a workspace. This function works with .csv, .xls and .xlsx format.

Usage

```
update_path(ws_xml_path, raw_data_path, pos_sap, pos_sa_item, verbose = TRUE)
```

Arguments

`ws_xml_path`	the path to the xml file of the workspace
`raw_data_path`	the new path to the raw data
`pos_sap`	the index of the SA-Processing containing the series (Optional)
`pos_sa_item`	the index of the SA-Item containing the series (Optional)
`verbose`	A boolean to print indications on the processing status (optional and TRUE by default)

Details

The argument `pos_sap` and `pos_sa_item` are optional. If `pos_sa_item` is not supplied, all SA-Item will be updated. If `pos_sap` is not supplied, all SA-Processing will be updated.

If `pos_sa_item` is supplied, `pos_sap` must be specified.

It's also important that the new data file has the same structure as the previous file : - same column names - same column position - same extension and format (.csv, .xls or .xlsx)

Value

the `workspace` ws_to augmented with series present in ws_from and not already in ws_to

Examples

```
library("RJDemetra")
new_dir <- tempdir()
ws_template_path <- file.path(system.file("extdata", package = "rjdworkspace"),
                         "WS")

# Moving the WS in a temporary environment
copy_ws(
    ws_name = "ws_example_path",
    from = ws_template_path,
    to = new_dir
)

# Moving the raw data in a temporary environment
data_path <- file.path(system.file("extdata", package = "rjdworkspace"),
                       "data_file.csv")
file.copy(
    from = data_path,
    to = new_dir
)

path_ws <- file.path(new_dir, "ws_example_path.xml")
new_raw_data_path <- file.path(new_dir, "data_file.csv")
```

```
update_path(
    ws_xml_path = path_ws,
    raw_data_path = new_raw_data_path,
    pos_sap = 1L,
    pos_sa_item = 1L:2L
)
update_path(
    ws_xml_path = path_ws,
    raw_data_path = new_raw_data_path,
    pos_sap = 1L
)
update_path(
    ws_xml_path = path_ws,
    raw_data_path = new_raw_data_path
)
```

Index