Package: rjd3workspace (via r-universe)

September 10, 2024

Type Package

Title Interface to 'JDemetra+ 3.x' time series analysis software.

Version 3.2.3.9000

Description R Interface to 'JDemetra+ 3.x' (<https://github.com/jdemetra>). It offers several functions to manipulate 'JDemetra+' workspaces, which can be read by the software and can store several seasonal adjusted series along with user-defined calendars or regression variables.

Depends R (>= 4.1.0),

Imports rJava (>= 1.0-6), rjd3toolkit (>= 3.2.2), rjd3tramoseats (>= 3.2.2), rjd3x13 (>= 3.2.2), rjd3providers (>= 3.2.2)

Remotes github::rjdverse/rjd3toolkit, github::rjdverse/rjd3tramoseats, github::rjdverse/rjd3x13, github::rjdverse/rjd3providers

SystemRequirements Java (>= 17)

License EUPL

URL https://github.com/rjdverse/rjd3workspace, https://rjdverse.github.io/rjd3workspace/

BugReports https://github.com/rjdverse/rjd3workspace/issues

Suggests knitr, rmarkdown,

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

Encoding UTF-8

Collate 'deprecated.R' 'utils.R' 'saitem.R' 'saprocessing.R' 'workspace.R' 'zzz.R'

Repository https://tanguybarthelemy.r-universe.dev

RemoteUrl https://github.com/rjdverse/rjd3workspace

RemoteRef HEAD

RemoteSha 19c72254580061d8d9c1771d7034db8eb38f480b

Contents

`.jsap_name` *Get the name of a SAProcessing or a SaItem*

Description

Functions to retrieve the name of a SAProcessing (`.jsap_name()`) or SaItem (`.jsa_name()`).

This function is used to retrieve the Java names of all the `sa_items` contained in a `SA-Processing`.

Usage

```
.jsa_name(jsa)

.jsap_name(jsap)

.jsap_sa_name(jsap)
```

Arguments

`jsap, jsa`	the object to retrieve the name from.
`jsap`	the java object representing the `SA-Processing`

Value

A vector `character`.

See Also

Other functions to retrieve the name of JDemetra+ objects (`workspace`, `SA-Processing` or `sa-item`): `.jsa_name`, `.jsap_name`.

Examples

```
y <- rjd3toolkit::ABS$X0.2.09.10.M

jws <- .jws_new()
jsap1 <- .jws_sap_new(jws, "sa1")

add_sa_item(jsap1, name = "x13", x = rjd3x13::x13(y))
add_sa_item(jsap1, name = "tramo", x = rjd3tramoseats::tramoseats(y))
add_sa_item(jsap1, name = "x13-2", x = y, rjd3x13::x13_spec())
add_sa_item(jsap1, name = "tramo-2", x = y, rjd3tramoseats::tramoseats_spec())

print(.jsap_sa_name(jsap1))
```

.jsa_metadata *Extract Java Metadata*

Description

Extract specific metadata or time series metadata of a SAItem.

Usage

```
.jsa_metadata(jsa, key)

.jsa_ts_metadata(jsa, key)
```

Arguments

jsa	Java SAItem object.
key	key of the metadata.

.jsa_read *Read SAItem*

Description

`.jsa_results()` extracts specific variables of the model of the SAItem while `.jsa_read()` extracts all the informations of a SAItem (see details).

Usage

```
.jsa_read(jsa)

.jsa_results(jsa, items = NULL)
```

Arguments

jsa	Java SAItem object.
items	vector of characters containing the variables to extract. See `rjd3x13::x13_dictionary()` or `rjd3tramoseats::tramoseats_dictionary()`. By default, extracts all the possible variables.

Details

A SAItem contains more information than just the results of a model. All those informations are extracted with the `.jsa_read()` function that returns a list with 5 objects:

- `ts`: the raw time series.
- `domainSpec`: initial specification. Reference for any relaxing of some elements of the specification.
- `estimationSpec`: specification used for the current estimation.
- `pointSpec`: specification corresponding to the results of the current estimation (fully identified model).
- `results`: the result of the model.

`.jws_new` *Create a workspace or a multi-processing*

Description

Functions to create a 'JDemetra+' workspace (`.jws_new()`) and to add a new multi-processing (`.jws_sap_new()`).

Usage

```
.jws_new(modelling_context = NULL)

.jws_sap_new(jws, name)
```

Arguments

`modelling_context`	The context (from `rjd3toolkit::modelling_context()`).
`jws`	A workspace object.
`name`	Character name of the new SAProcessing.

Examples

```
# To create an empty 'JDemetra+' workspace
jwk <- .jws_new()
jsap <- .jws_sap_new(jwk, "sa1")
```

.jws_open *Load a 'JDemetra+' workpace*

Description

`.jws_open()` loads a workspace and `.jws_compute()` computes it (to be able to get all the models).

Usage

```
.jws_open(file)

.jws_load(file)
```

Arguments

`file`	the path to the 'JDemetra+' workspace to load. By default a dialog box opens.

See Also

`read_workspace()` to import all the models of a workspace.

.jws_sap *Extract a SAProcessing or a SaItem*

Description

Extract a SAProcessing or a SaItem

Usage

```
.jsap_sa(jsap, idx)

.jws_sap(jws, idx)
```

Arguments

`idx`	index of the object to extract.
`jws, jsap`	the workspace or the SAProcessing.

.jws_sap_count *Count the number of objects inside a workspace or SAProcessing*

Description

Functions to count the number of SAProcessing inside a workspace (`jws_sap_count`) or the number of SaItem inside a SAProcessing (`jsap_sa_count`).

Usage

```
.jsap_sa_count(jsap)

.jws_sap_count(jws)
```

Arguments

`jws`, `jsap`	the workspace or the SAProcessing.

add_calendar *Add Calendar to Workspace*

Description

Add Calendar to Workspace

Usage

```
add_calendar(jws, name, calendar)
```

Arguments

`jws`	A workspace object.
`name`	the name of the calendar to add.
`calendar`	the calendar to add.

add_sa_item *Add SAItem to SAProcessing*

Description

Add SAItem to SAProcessing

Usage

```
add_sa_item(jsap, name, x, spec, ...)
```

Arguments

jsap	the SAProcessing.
name	the name of SAItem.
x	either a seasonal adjustment model (from rjd3x13::x13() or rjd3tramoseats::tramoseats()), a SaItem or a "ts" object.
spec	the specification to use when x is a "ts" object.
...	other unused parameters.

Examples

```
dir <- tempdir()
y <- rjd3toolkit::ABS$X0.2.09.10.M
jws <- .jws_new()
jsap1 <- .jws_sap_new(jws, "sa1")
add_sa_item(jsap1, name = "x13", x = rjd3x13::x13(y))
add_sa_item(jsap1, name = "tramo", x = rjd3tramoseats::tramoseats(y))
add_sa_item(jsap1, name = "x13-2", x = y, rjd3x13::x13_spec())
add_sa_item(jsap1, name = "tramo-2", x = y, rjd3tramoseats::tramoseats_spec())
save_workspace(jws, file.path(dir, "workspace.xml"))
```

add_variable *Add Variable to Workspace*

Description

Add Variable to Workspace

Usage

```
add_variable(jws, group, name, y)
```

Arguments

jws	A workspace object.
group, name	the group and the name of the variable to add.
y	the variable (a ts object).

---

deprecated-rjd3workspace *Deprecated functions*

Description

Deprecated functions

Usage

```
.jmp_sa_count(jmp)

.jmp_name(jmp)

.jmp_sa(jmp, idx)

.jmp_sa_name(jmp)

.jmp_load(jmp)

.jws_multiprocessing(jws, idx)

.jws_multiprocessing_new(jws, name)

.jws_multiprocessing_count(jws)
```

Arguments

jmp, idx, jws, name	Parameters.

---

get_context *Get Context from Workspace*

Description

Get Context from Workspace

Usage

```
get_context(jws)
```

Arguments

jws	the workspace.

---

make_copy *Copy Workspace or a SAProcessing*

Description

Copy Workspace or a SAProcessing

Usage

```
.jsap_make_copy(jsap)

.jws_make_copy(jws)
```

Arguments

jws, jsap	Java Workspace or Multiprocessing

---

read_calendars *Title*

Description

Title

Usage

```
read_calendars(file)
```

Arguments

file	

read_variables *Title*

Description

Title

Usage

```
read_variables(file)
```

Arguments

file

read_workspace *Read all SaItems*

Description

Functions to read all the SAItem of a SAProcessing (`read_sap()`) or a workspace (`read_workspace()`).

Usage

```
read_sap(jsap)

read_workspace(jws, compute = TRUE)
```

Arguments

jsap	Java SAProcessing.
jws	Java workspace.
compute	Compute the workspace.

Examples

```
file<-system.file("workspaces", "test.xml", package = "rjd3workspace")
jws<-.jws_load(file)
# We don't compute the workspace
rws<-read_workspace(jws, FALSE)
```

refresh *Refresh Workspace or SAProcessing*

Description

Refresh Workspace or SAProcessing

Usage

```
.jsap_refresh(
  jsap,
 policy = c("FreeParameters", "Complete", "Outliers_StochasticComponent", "Outliers",
    "FixedParameters", "FixedAutoRegressiveParameters", "Fixed"),
  period = 0,
  start = NULL,
  end = NULL,
  info = c("All", "Data", "None")
)

.jws_refresh(
  jws,
 policy = c("FreeParameters", "Complete", "Outliers_StochasticComponent", "Outliers",
    "FixedParameters", "FixedAutoRegressiveParameters", "Fixed"),
  period = 0,
  start = NULL,
  end = NULL,
  info = c("All", "Data", "None")
)
```

Arguments

policy	the refresh policy to apply (see details).
period, start, end	to specify the span on which outliers will not be re-identified (i.e.: re-detected) when policy = "Outliers" or policy = "Outliers_StochasticComponent". Span definition: period: numeric, number of observations in a year (12, 4...). start and end: first and last date from which outliers will not be re-identfied, defined as arrays of two elements: year and first period (for example, if period = 12, c(1980, 1) for January 1980). If they are not specified, the outliers will be re-identified on the whole series.
info	information to refresh.
jws, jsap	Java Workspace or Multiprocessing

Details

Available refresh policies are:

Current: applying the current pre-adjustment reg-arima model and adding the new raw data points as Additive Outliers (defined as new intervention variables)

Fixed: applying the current pre-adjustment reg-arima model and replacing forecasts by new raw data points.

FixedParameters: pre-adjustment reg-arima model is partially modified: regression coefficients will be re-estimated but regression variables, Arima orders and coefficients are unchanged.

FixedAutoRegressiveParameters: same as FixedParameters but Arima Moving Average coefficients (MA) are also re-estimated, Auto-regressive (AR) coefficients are kept fixed.

FreeParameters: all regression and Arima model coefficients are re-estimated, regression variables and Arima orders are kept fixed.

Outliers: regression variables and Arima orders are kept fixed, but outliers will be re-detected on the defined span, thus all regression and Arima model coefficients are re-estimated

Outliers_StochasticComponent: same as "Outliers" but Arima model orders (p,d,q)(P,D,Q) can also be re-identified.

`regarima_read_spec` *Title*

Description

Title

Usage

```
regarima_read_spec(file)
```

Arguments

`file`

`regarima_write_spec` *Title*

Description

Title

Usage

```
regarima_write_spec(spec, file)
```

Arguments

`file`

replace_sa_item *Replace or Remove a SaItem*

Description

replace_sa_item() replaces a SaItem of a SAProcessing and remove_sa_item() removes a SaItem from a SAProcessing

This functions clear a SA-Processing by removing all the sa-item contained.

Usage

```
replace_sa_item(jsap, idx, jsa)

remove_sa_item(jsap, idx)

remove_all_sa_item(jsap)

transfer_series(jsap_from, jsap_to, selected_series, print_indications = TRUE)
```

Arguments

jsap	the SAProcessing to modify.
idx	index of the target SaItem.
jsa	the new SaItem.
jsap_from	The SA-Processing from which to take the series
jsap_to	The SA-Processing in which to paste the series
selected_series	The vector containing the series-to-update's names.
print_indications	A boolean to print indications on the processing status (optional)

Details

If selected_series is missing, all series from jsap_from will be copied. In this context, the word series refers to sa-item.

Value

NULL returned invisibly

NULL returned invisibly

save_workspace *Save Workspace*

Description

Save Workspace

Usage

```
save_workspace(jws, file, replace = FALSE)
```

Arguments

jws	the workspace object to export.
file	the path where to export the 'JDemetra+' workspace (.xml file).
replace	boolean indicating if the workspace should be replaced if it already exists.

Examples

```
dir <- tempdir()
jws <- .jws_new()
jsap1 <- .jws_sap_new(jws, "sa1")
y <- rjd3toolkit::ABS$X0.2.09.10.M
add_sa_item(jsap1, name = "x13", x = y, rjd3x13::x13_spec())
save_workspace(jws, file.path(dir, "workspace.xml"))
```

set_comment *Get/Set SaItem Comment*

Description

Get/Set SaItem Comment

Usage

```
set_comment(jsap, idx, comment)

get_comment(jsa)
```

Arguments

jsap	the SAProcessing to modify.
idx	index of the target SaItem.
comment	character containing the comment.
jsa	a SaItem.

set_context *Set Context of a Workspace*

Description

Set Context of a Workspace

Usage

```
set_context(jws, modelling_context = NULL)
```

Arguments

jws	A workspace object.
modelling_context	The context (from rjd3toolkit::modelling_context()).

set_name *Set the name associated to a SaItem Comment*

Description

Set the name associated to a SaItem Comment

Usage

```
set_name(jsap, idx, name)
```

Arguments

jsap	the SAProcessing to modify.
idx	index of the target SaItem.
name	character containing the name of the SAItem.

See Also

.jsa_name()

set_priority *Get/Set SaItem Priority*

Description

Get/Set SaItem Priority

Usage

```
set_priority(jsap, idx, priority = 0)

get_priority(jsa)
```

Arguments

jsap	the SAProcessing to modify.
idx	index of the target SaItem.
priority	integer containing the priority.
jsa	a SaItem.

set_raw_data *Get/Set the Raw Data of a SaItem*

Description

Get/Set the Raw Data of a SaItem

Usage

```
set_raw_data(jsap, idx, y)

get_raw_data(jsa)
```

Arguments

jsap	the SAProcessing to modify.
idx	index of the target SaItem.
y	the new raw time serie.
jsa	a SaItem.

set_specification *Set Specification or Data of a SaItem*

Description

Set Specification or Data of a SaItem

Usage

```
set_specification(jsap, idx, spec)

set_domain_specification(jsap, idx, spec)
```

Arguments

jsap	the SAProcessing to modify.
idx	index of the target SaItem.
spec	the new specification.

set_ts *Get/Set the time series of a SaItem*

Description

Get/Set the time series of a SaItem

Usage

```
set_ts(jsap, idx, y)

get_ts(jsa)
```

Arguments

jsap	the SAProcessing to modify.
idx	index of the target SaItem.
y	a "full" time series (jd3-like).
jsa	a SaItem.

set_ts_metadata *Set Time Series Metadata of a SaItem*

Description

Function to set the time series metadata of a SaItem (provider, source of the data...). `set_ts_metadata()` uses the metadata of another SaItem while `put_ts_metadata()` allows to update a specific key with a new information.

Usage

```
set_ts_metadata(jsap, idx, ref_jsa)

put_ts_metadata(jsap, idx, key, value)
```

Arguments

`jsap`	the SAProcessing to modify.
`idx`	index of the target SaItem.
`ref_jsa`	a reference SaItem containing the metadata.
`key`	key of the metadata.
`value`	value of the metadata.

Examples

```
# Change the file of a given item
file <- system.file("workspaces", "test.xml", package = "rjd3workspace")
jws <- .jws_load(file)
jsap <- .jws_sap(jws, 1)
jsa <- .jsap_sa(jsap, 1)
nid <- rjd3providers::spreadsheet_change_file(.jsa_ts_metadata(jsa, "@id"), "test.xlsx")
put_ts_metadata(jsap, 1, "@id", nid)
jsa <- .jsap_sa(jsap, 1)
.jsa_ts_metadata(jsa, "@id")
```

tramoseats_read_spec *Title*

Description

Title

Usage

```
tramoseats_read_spec(file)
```

Arguments

file

---

tramoseats_write_spec *Title*

Description

Title

Usage

```
tramoseats_write_spec(spec, file)
```

Arguments

file

---

tramo_read_spec *Title*

Description

Title

Usage

```
tramo_read_spec(file)
```

Arguments

file

---

tramo_write_spec *Title*

Description

Title

Usage

```
tramo_write_spec(spec, file)
```

Arguments

file

write_calendars *Title*

Description

Title

Usage

```
write_calendars(calendars, file)
```

Arguments

```
file
```

write_variables *Title*

Description

Title

Usage

```
write_variables(vars, file)
```

Arguments

```
file
```

x13_read_spec *Title*

Description

Title

Usage

```
x13_read_spec(file)
```

Arguments

```
file
```

x13_write_spec *Title*

Description

Title

Usage

```
x13_write_spec(spec, file)
```

Arguments

file

Index